A

US006321338B1

(12) **United States Patent**
Porras et al.

(10) **Patent No.: US 6,321,338 B1**
(45) **Date of Patent: Nov. 20, 2001**

(54) **NETWORK SURVEILLANCE**

(75) Inventors: **Phillip A. Porras**, Mountain View; **Alfonso Valdes**, San Carlos, both of CA (US)

(73) Assignee: **SRI International**, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/188,739**

(22) Filed: **Nov. 9, 1998**

(51) **Int. Cl.**[7] **G06F 11/30**; G06F 12/14
(52) **U.S. Cl.** **713/201**; 709/224
(58) **Field of Search** 713/201; 709/224

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,672,609	6/1987	Humphrey et al.	371/21
4,773,028	9/1988	Tallman	364/550
5,210,704	5/1993	Husseiny	364/551.01
5,539,659 *	7/1996	McKee et al.	709/224
5,557,742	9/1996	Smaha et al.	395/186
5,706,210 *	1/1998	Kumano et al.	709/224
5,790,799 *	8/1998	Mogul	709/224
5,974,237 *	10/1999	Shurmer et al.	709/224
6,009,467 *	12/1999	Ratcliff et al.	709/224

OTHER PUBLICATIONS

Debar et al., "A Neural Network Component for an Intrusion Detection System," © 1992 IEEE.
Denning et al., "Prototype IDES: A Real–Time Intrusion–Detection Expert System," SRI Project ECU 7508, SRI International, MenloPark, California, Aug. 1987.
Denning et al., "Requirements and Model For IDES—A Real–Time Intrusion–Detection Expert System," SRI Project 6169, SRI International, Menlo Park, CA, Aug. 1985.
Denning, "An Intrusion–Detection Model," SRI International, Menlo Park, CA, Technical Report CSL–149, Nov. 1985.
Dowell, "The Computerwatch Data Reduction Tool," AT&T Bell Laboratories, Whippany, New Jersey.
Fox et al., "A Neural Network Approach Towards Intrusion Detection," Harris Corporation, Government Information Systems Division, Melbourne, FL, Jul. 2, 1990.
Garvey et al., "Model–Based Intrusion Detection," Proceedings of the 14th National Computer Security Conference, Washington, DC, Oct. 1991.
Ilgun et al., State Transition Analysis: A Rule–Based Intrusion Detection Approach, IEEE Transactions on Software Engineering, vol. 21, No. 3, Mar. 1995.
Javitz et al., "The SRI IDES Statistical Anomaly Detector," Proceedings, 1991 IEEE Symposium on Security and Privacy, Oakland, California, May 1991.
Liepins et al., "Anomaly Detection: Purpose and Framework," US DOE Office of Safeguards and Security.
Lunt et al., "An Expert System to Classify and Sanitize Text," SRI International, Computer Science Laboratory, Menlo Park, CA.

(List continued on next page.)

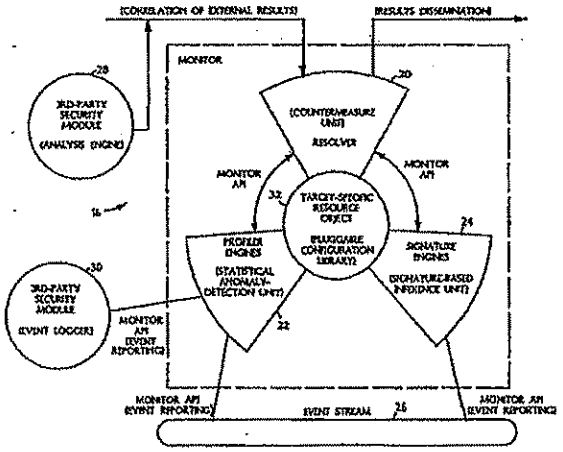
Primary Examiner—Thomas M. Heckler
(74) *Attorney, Agent, or Firm*—Fish & Richardson P.C.

(57) **ABSTRACT**

A method of network surveillance includes receiving network packets handled by a network entity and building at least one long-term and a least one short-term statistical profile from a measure of the network packets that monitors data transfers, errors, or network connections. A comparison of the statistical profiles is used to determine whether the difference between the statistical profiles indicates suspicious network activity.

27 Claims, 5 Drawing Sheets

Microfiche Appendix Included
(10 Microfiche, 956 Pages)



OTHER PUBLICATIONS

Lunt, "A Survey of Intrusion Detection Techniques," Computers & Security, 12 (1993) 405–418.
Lunt, "Automated Audit Trail Analysis and Intrusion Detection: A Survey," Proceedings of the 11th National Computer Security Conference, Baltimore, MD, Oct. 1988.
Lunt et al, "Knowledge–Based Intrusion Detection".
Lunt et al., "A Prototype Real–Time Intrusion–Detection Expert System," Proceedings of the 1988 IEEE Symposium on Security and Privacy, Apr. 1988.
Porras et al., EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances, 20th NISSC—Oct. 9, 1997.
Porras et al., Penetration State Transition Analysis A Rule–Based Intrusion Detection Approach, © 1992 IEEE.
Sebring et al., Expert Systems in Intrusion Detection: A Case Study.
Shieh et al., A Pattern–Oriented Intrusion–Detection Model and Its Applications © 1991 IEEE.
Smaha, "Haystack: An Intrusion Detection System," © 1988 IEEE Computer Society Press: Proceedings of the Fourth Aerospace Computer Security Applications Conference, 1988, pp. 37–44.
Snapp, "Signature Analysis and Communication Issues in a Distributed Intrusion Detection System," Thesis 1991.
Snapp et al., "DIDS (Distributed Intrusion Detection System)—Motivation, Architecture, and An Early Protoype, "Computer Security Laboratory, Division of Computer Science, Univ. of California, Davis, Davis, CA.
Tener, "AI & 4GL: Automated Detection and Investigation Tools," Computer Security in the Age of Information, Proceedings of the Fifth IFIP International Conference on Computer Security, W.J. Caelli (ed.).
Teng et al., "Adaptive Real–Time Anomaly Detection Using Inductively Generated Sequential Patterns," © 1990.
Vaccaro et al., "Detection of Anomalous Computer Session Activity," © 1989 IEEE.
Weiss, "Analysis of Audit and Protocol Data using Methods from Artificial Intelligence," Siemens AG, Munich, West Germany.
Winkler, "A UNIX Prototype for Intrusion and Anomaly Detection in Secure Networks," © Planning Research Corp. 1990.
Jarvis et al., The NIDES Statistical Component Description and Justification, SRI International Annual Report A010, Mar. 7, 1994.
Debar, et al., "Towards a Taxonomy of Intrusion–Detection Systems," Computers Networks 31 (1999), 805–822.
Garvey, et al., "An Inference Technique for Integrating Knowledge from Disparate Sources," Proc. IJCAI, Vancouver, B.C., Aug., 1981, 319–325.
Kaven, "The Digital Doorman," PC Magazine, Nov. 16, 1999.
Lindqvist, et al., "Detecting Computer and Network Misuse Through the Production–Based Expert System Toolset (P–BEST)," Oct. 25, 1998.

* cited by examiner

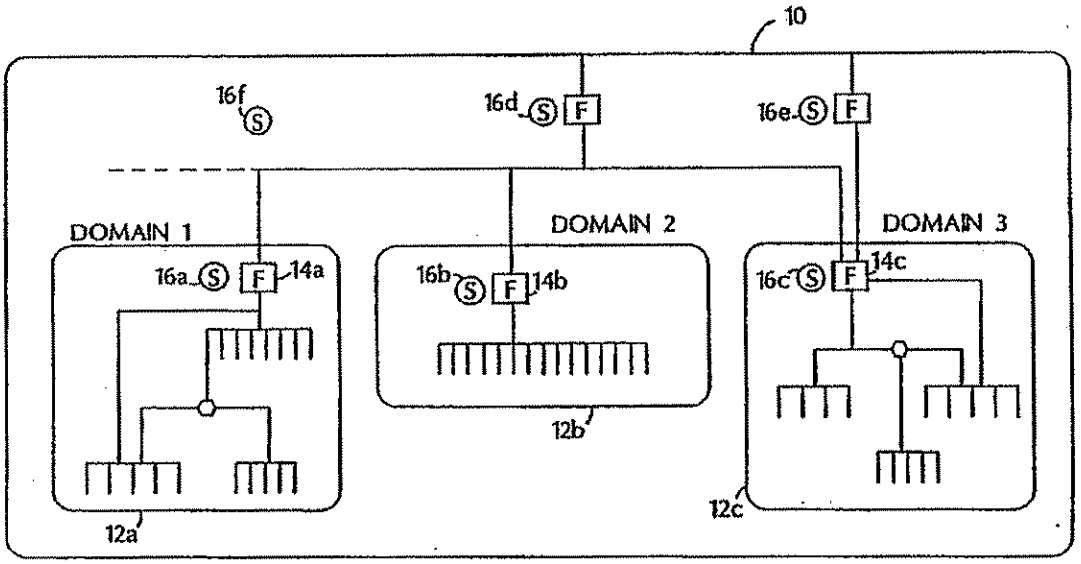

10
16f
S
16d
S
F
16e
S
F
DOMAIN 1
DOMAIN 2
DOMAIN 3
16a
S
F
14a
16b
S
F
14b
16c
S
F
14c
12a
12b
12c

FIG. 1

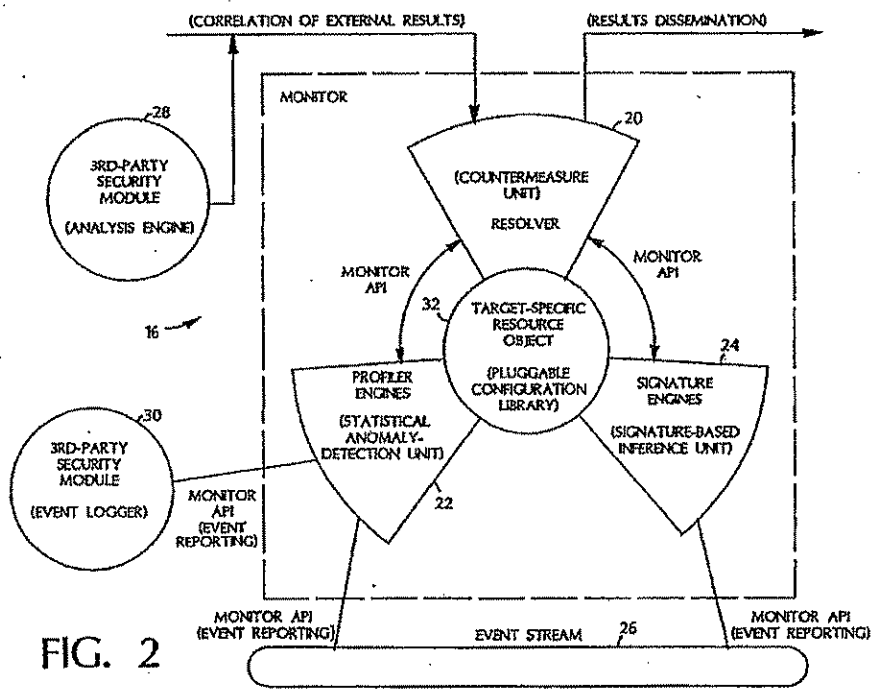

(CORRELATION OF EXTERNAL RESULTS)
(RESULTS DISSEMINATION)
MONITOR
28
3RD-PARTY SECURITY MODULE
(ANALYSIS ENGINE)
20
(COUNTERMEASURE UNIT)
RESOLVER
MONITOR API
MONITOR API
32
TARGET-SPECIFIC RESOURCE OBJECT
(PLUGGABLE CONFIGURATION LIBRARY)
16
24
PROFILER ENGINES
(STATISTICAL ANOMALY-DETECTION UNIT)
SIGNATURE ENGINES
(SIGNATURE-BASED INFERENCE UNIT)
30
3RD-PARTY SECURITY MODULE
(EVENT LOGGER)
MONITOR API (EVENT REPORTING)
22
MONITOR API (EVENT REPORTING)
EVENT STREAM
26
MONITOR API (EVENT REPORTING)

FIG. 2

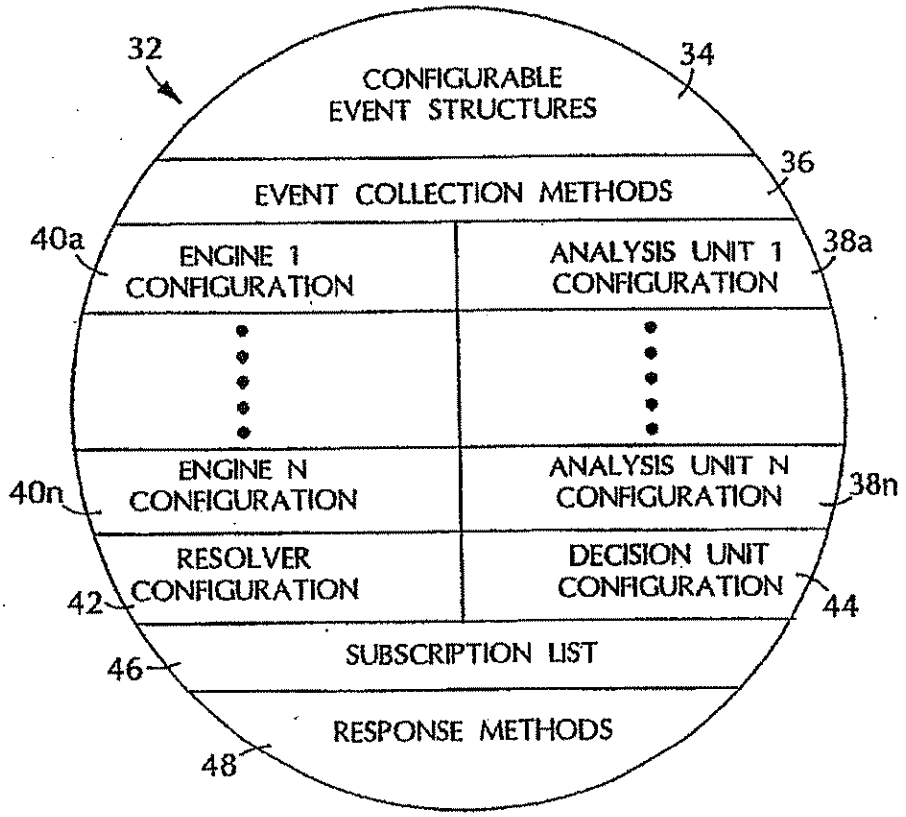

32
34
CONFIGURABLE EVENT STRUCTURES
36
EVENT COLLECTION METHODS
40a
ENGINE 1 CONFIGURATION
ANALYSIS UNIT 1 CONFIGURATION
38a
ENGINE N CONFIGURATION
40n
ANALYSIS UNIT N CONFIGURATION
38n
RESOLVER CONFIGURATION
42
DECISION UNIT CONFIGURATION
44
46
SUBSCRIPTION LIST
RESPONSE METHODS
48

FIG. 3

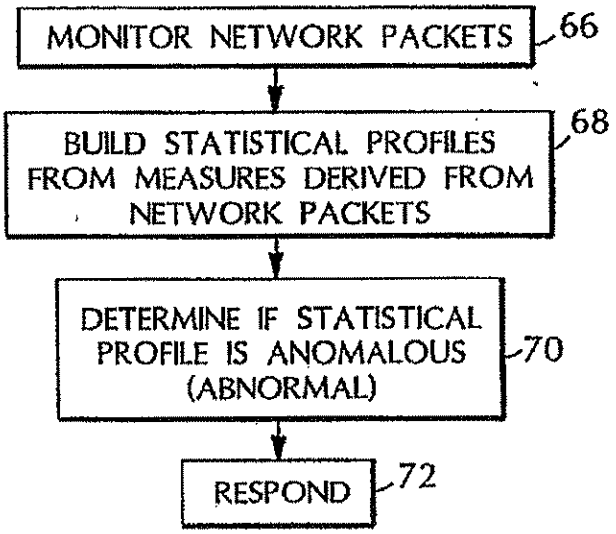

MONITOR NETWORK PACKETS
66
BUILD STATISTICAL PROFILES FROM MEASURES DERIVED FROM NETWORK PACKETS
68
DETERMINE IF STATISTICAL PROFILE IS ANOMALOUS (ABNORMAL)
70
RESPOND
72

FIG. 4

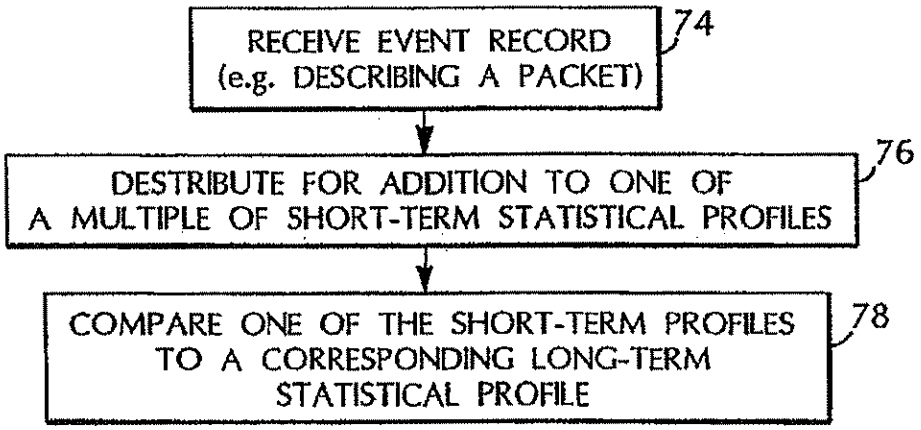

RECEIVE EVENT RECORD (e.g. DESCRIBING A PACKET)
74
DESTRIBUTE FOR ADDITION TO ONE OF A MULTIPLE OF SHORT-TERM STATISTICAL PROFILES
76
COMPARE ONE OF THE SHORT-TERM PROFILES TO A CORRESPONDING LONG-TERM STATISTICAL PROFILE
78

FIG. 5

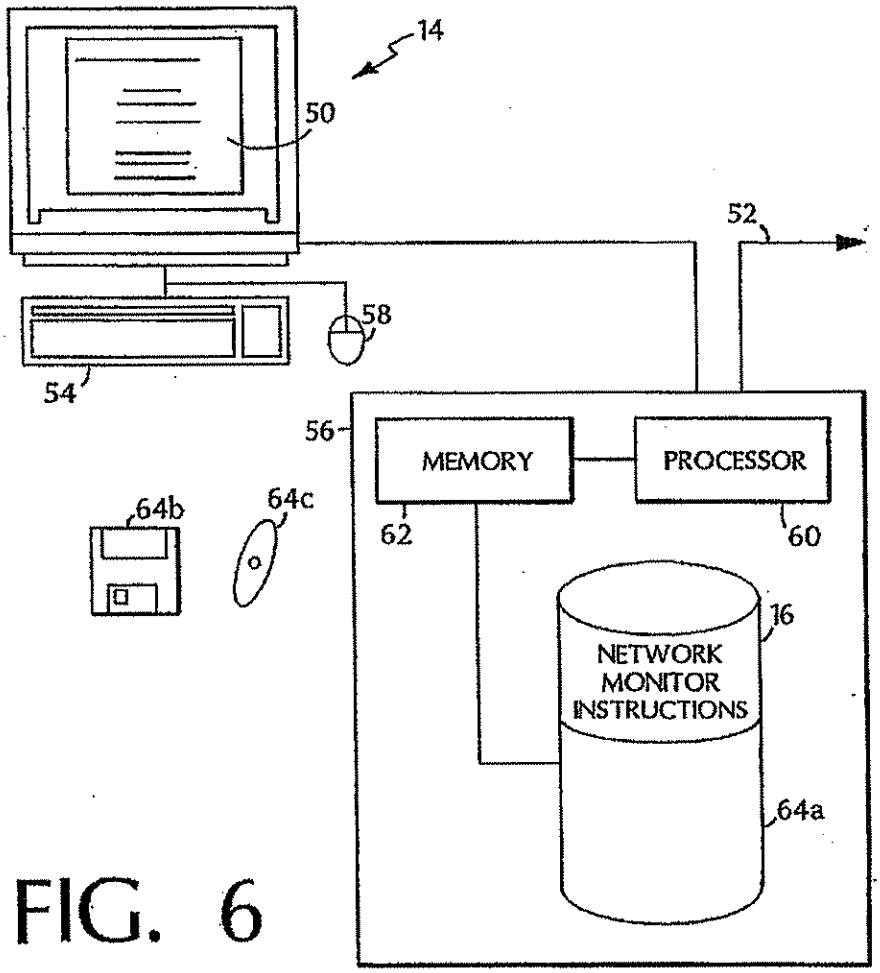

14
50
52
58
54
56
MEMORY
PROCESSOR
62
60
64b
64c
16
NETWORK
MONITOR
INSTRUCTIONS
64a

FIG. 6

NETWORK SURVEILLANCE

REFERENCE TO GOVERNMENT FUNDING

This invention was made with Government support under Contract Number F30602-96-C-0294 awarded by DARPA. The Government has certain rights in this invention.

REFERENCE TO APPENDIX

A microfiche appendix is included as part of the specification. The microfiche appendix includes material subject to copyright protection. The copyright owner does not object to the reproduction of the microfiche appendix, as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights. This application contains Microfiche Appendix containing ten (10) slides and 956 frames.

BACKGROUND

The invention relates to computer networks.

Computer networks offer users ease and efficiency in exchanging information. Networks tend to include conglomerates of integrated commercial and custom-made components, interoperating and sharing information at increasing levels of demand and capacity. Such varying networks manage a growing list of needs including transportation, commerce, energy management, communications, and defense.

Unfortunately, the very interoperability and sophisticated integration of technology that make networks such valuable assets also make them vulnerable to attack, and make dependence on networks a potential liability. Numerous examples of planned network attacks, such as the Internet worm, have shown how interconnectivity can be used to spread harmful program code. Accidental outages such as the 1980 ARPAnet collapse and the 1990 AT&T collapse illustrate how seemingly localized triggering events can have globally disastrous effects on widely distributed systems. In addition, organized groups have performed malicious and coordinated attacks against various online targets.

SUMMARY

In general, in one aspect, a method of network surveillance includes receiving network packets (e.g., TCP/IP packets) handled by a network entity and building at least one long-term and at least one short-term statistical profile from at least one measure of the network packets that monitors data transfers, errors, or network connections. A comparison of at least one long-term and at least one short-term statistical profile is used to determine whether the difference between the short-term statistical profile and the long-term statistical profile indicates suspicious network activity.

Embodiments may include one or more of the following features. The measure may monitor data transfers by monitoring network packet data transfer commands, data transfer errors, and/or monitoring network packet data transfer volume. The measure may monitor network connections by monitoring network connection requests, network connection denials, and/or a correlation of network connections requests and network connection denials. The measure may monitor errors by monitoring error codes included in a network packet such as privilege error codes and/or error codes indicating a reason a packet was rejected.

The method may also include responding based on the determining whether the difference between a short-term statistical profile and a long-term statistical profile indicates suspicious network activity. A response may include altering analysis of network packets and/or severing a communication channel. A response may include transmitting an event record to a network monitor, such as hierarchically higher network monitor and/or a network monitor that receives event records from multiple network monitors.

The network entity may be a gateway, a router, or a proxy server. The network entity may instead be a virtual private network entity (e.g., node).

In general, in another aspect, a method of network surveillance includes monitoring network packets handled by a network entity and building a long-term and multiple short-term statistical profiles of the network packets. A comparison of one of the multiple short-term statistical profiles with the long-term statistical profile is used to determine whether the difference between the short-term statistical profiles and the long-term statistical profile indicates suspicious network activity.

Embodiments may include one or more of the following. The multiple short-term statistical profiles may monitor different anonymous FTP sessions. Building multiple short-term statistical profiles may include deinterleaving packets to identify a short-term statistical profile.

In general, in another aspect, a computer program product, disposed on a computer readable medium, includes instructions for causing a processor to receive network packets handled by a network entity and to build at least one long-term and at least one short-term statistical profile from at least one measure of the network packets that monitors data transfers, errors, or network connections. The instructions compare a short-term and a long-term statistical profile to determine whether the difference between the short-term statistical profile and the long-term statistical profile indicates suspicious network activity.

In general, in another aspect, a method of network surveillance includes receiving packets at a virtual private network entity and statistically analyzing the received packets to determine whether the packets indicate suspicious network activity. The packets may or may not be decrypted before statistical analysis.

Advantages may include one or more of the following. Using long-term and a short-term statistical profiles from measures that monitor data transfers, errors, or network connections protects network components from intrusion. As long-term profiles represent "normal" activity, abnormal activity may be detected without requiring an administrator to catalog each possible attack upon a network. Additionally, the ability to deinterleave packets to create multiple short-term profiles for comparison against a long-term profile enables the system to detect abnormal behavior that may be statistically ameliorated if only a single short-term profile was created.

The scheme of communication network monitors also protects networks from more global attacks. For example, an attack made upon one network entity may cause other entities to be alerted. Further, a monitor that collects event reports from different monitors may correlate activity to identify attacks causing disturbances in more than one network entity.

Additionally, statistical analysis of packets handled by a virtual private network enable detection of suspicious network activity despite virtual private network security techniques such as encryption of the network packets.

Other features and advantages will become apparent from the following description, including the drawings, and from the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a diagram of network monitors deployed in an enterprise.

FIG. 2 is a diagram of a network monitor that monitors an event stream.

FIG. 3 is a diagram of a resource object that configures the network monitor of FIG. 2.

FIG. 4 is a flowchart illustrating network surveillance.

FIG. 5 is a flowchart illustrating multiple short-term statistical profiles for comparison against a single long-term statistical profile.

FIG. 6 is a diagram of a computer platform suitable for deployment of a network monitor.

DETAILED DESCRIPTION

Referring to FIG. 1, an enterprise **10** includes different domains **12***a*–**12***c*. Each domain **12***a*–**12***c* includes one or more computers offering local and network services that provide an interface for requests internal and external to the domain **12***a*–**12***c*. Network services include features common to many network operating systems such as mail, HTTP, FTP, remote login, network file systems, finger, Kerberos, and SNMP. Some domains **12***a*–**12***c* may share trust relationships with other domains (either peer-to-peer or hierarchical). Alternatively, domains **12***a*–**12***c* may operate in complete mistrust of all others, providing outgoing connections only or severely restricting incoming connections. Users may be local to a single domain or may possess accounts on multiple domains that allow them to freely establish connections throughout the enterprise **10**.

As shown, the enterprise **10** includes dynamically deployed network monitors **16***a*–**16***f* that analyze and respond to network activity and can interoperate to form an analysis hierarchy. The analysis hierarchy provides a framework for the recognition of more global threats to interdomain connectivity, including coordinated attempts to infiltrate or destroy connectivity across an entire network enterprise **10**. The hierarchy includes service monitors **16***a*–**16***c*, domain monitors **16***d*–**16***e*, and enterprise monitors **16***f*.

Service monitors **16***a*–**16***c* provide local real-time analysis of network packets (e.g., TCP/IP packets) handled by a network entity **14***a*–**14***c*. Network entities include gateways, routers, firewalls, or proxy servers. A network entity may also be part of a virtual private network. A virtual private network (VPN) is constructed by using public wires to connect nodes. For example, a network could use the Internet as the medium for transporting data and use encryption and other security mechanisms to ensure that only authorized users access the network and that the data cannot be intercepted. A monitor **16***a*–**16***f* can analyze packets both before and after decryption by a node of the virtual private network.

Information gathered by a service monitor **16***a*–**16***c* can be disseminated to other monitors **16***a*–**16***f*, for example, via a subscription-based communication scheme. In a subscription-based scheme client monitors subscribe to receive analysis reports produced by server monitors. As a monitor **16***a*–**16***f* produces analysis reports, the monitor **16***a*–**16***f* disseminates these reports asynchronously to subscribers. Through subscription, monitors **16***a*–**16***f* distributed throughout a large network are able to efficiently disseminate reports of malicious activity without requiring the overhead of synchronous polling.

Domain monitors **16***d*–**16***e* perform surveillance over all or part of a domain **12***a*–**12***c*. Domain monitors **16***d*–**16***e* correlate intrusion reports disseminated by individual service monitors **16***a*–**16***c*, providing a domain-wide perspective of activity (or patterns of activity). In addition to domain surveillance, domain monitors **16***a*–**16***c* can reconfigure system parameters, interface with other monitors beyond a domain, and report threats against a domain **12***a*–**12***c* to administrators. Domain monitors **16***d*–**16***e* can subscribe to service monitors **16***a*–**16***c*. Where mutual trust among domains **12***a*–**12***c* exists, domain monitors **16***d*–**16***e* may establish peer relationships with one another. Peer-to-peer subscription allows domain monitors **16***d*–**16***e* to share analysis reports produced in other domains **12***a*–**12***c*. Domain monitors **16***d*–**16***e* may use such reports to dynamically sensitize their local service monitors **16***a*–**16***c* to malicious activity found to be occurring outside a domain **12***a*–**12***c*. Domain monitors **16***d*–**16***e* may also operate within an enterprise hierarchy where they disseminate analysis reports to enterprise monitors **16***f* for global correlation.

Enterprise monitors **16***f* correlate activity reports produced across the set of monitored domains **12***a*–**12***c*. Enterprise **10** surveillance may be used where domains **12***a*–**12***c* are interconnected under the control of a single organization, such as a large privately owned WAN (Wide Area Network). The enterprise **10**, however, need not be stable in its configuration or centrally administered. For example, the enterprise **10** may exist as an emergent entity through new interconnections of domains **12***a*–**12***c*. Enterprise **10** surveillance is very similar to domain **12***a*–**12***c* surveillance: an enterprise monitor **16***f* subscribes to various domain monitors **16***d*–**16***e*, just as the domain monitors **16***d*–**16***e* subscribed to various service monitors **16***a*–**16***c*. The enterprise monitor **16***f* (or monitors, as it would be important to avoid centralizing any analysis) focuses on network-wide threats such as Internet worm-like attacks, attacks repeated against common network services across domains, or coordinated attacks from multiple domains against a single domain. As an enterprise monitor **16***f* recognizes commonalities in intrusion reports across domains (e.g., the spreading of a worm or a mail system attack repeated throughout the enterprise **10**), the monitor **16***f* can help domains **12***a*–**12***c* counter the attack and can sensitize other domains **12***a*–**12***c* to such attacks before they are affected. Through correlation and sharing of analysis reports, reports of problems found by one monitor **16***a*–**16***f* may propagate to other monitors **16***a*–**16***f* throughout the network. Interdomain event analysis is vital to addressing more global, information attacks against the entire enterprise **10**.

Referring to FIG. 2, each monitor **16** includes one or more analysis engines **22**, **24**. These engines **22**, **24** can be dynamically added, deleted, and modified as necessary. In the dual-analysis configuration shown, a monitor **16** instantiation includes a signature analysis engine **22** and a statistical profiling engine **24**. In general, a monitor **16** may include additional analysis engines that may implement other forms of analysis. A monitor **16** also includes a resolver **20** that implements a response policy and a resource object **32** that configures the monitor **16**. The monitors **16** incorporate an application programmers' interface (API) that enhances encapsulation of monitor functions and eases integration of third-party intrusion-detection tools **28**, **30**.

Each monitor **16** can analyze event records that form an event stream. The event stream may be derived from a variety of sources such as TCP/IP network packet contents or event records containing analysis reports disseminated by other monitors. For example, an event record can be formed from data included in the header and data segment of a network packet. The volume of packets transmitted and

received, however, dictates careful assessment of ways to select and organize network packet information into event record streams.

Selection of packets can be based on different criteria. Streams of event records can be derived from discarded traffic (i.e., packets not allowed through the gateway because they violate filtering rules), pass-through traffic (i.e., packets allowed into the internal network from external sources), packets having a common protocol (e.g., all ICMP (Internet Control Message Protocol) packets that reach the gateway), packets involving network connection management (e.g., SYN, RESET, ACK, [window resize]), and packets targeting ports to which an administrator has not assigned any network service and that also remain unblocked by the firewall. Event streams may also be based on packet source addresses (e.g., packets whose source addresses match well-known external sites such as satellite offices or have raised suspicion from other monitoring efforts) or destination addresses (e.g., packets whose destination addresses match a given internal host or workstation). Selection can also implement application-layer monitoring (e.g., packets targeting a particular network service or application). Event records can also be produced from other sources of network packet information such as report logs produced by network entities. Event streams can be of very fine granularity. For example, a different stream might be derived for commands received from different commercial web-browsers since each web-browser produces different characteristic network activity.

A monitor 16 can also construct interval summary event records, which contain accumulated network traffic statistics (e.g., number of packets and number of kilobytes transferred). These event records are constructed at the end of each interval (e.g., once per N seconds). Event records are forwarded to the analysis engines 22, 24 for analysis.

The profile engine 22 can use a wide range of multivariate statistical measures to profile network activity indicated by an event stream. A statistical score represents how closely currently observed usage corresponds to the established patterns of usage. The profiler engine 22 separates profile management and the mathematical algorithms used to assess the anomaly of events. The profile engine 22 may use a statistical analysis technique described in A. Valdes and D. Anderson, "Statistical Methods for Computer Usage Anomaly Detection Using NIDES", Proceedings of the Third International Workshop on Rough Sets and Soft Computing, January 1995, which is incorporated by reference in its entirety. Such an engine 22 can profile network activity via one or more variables called measures. Measures can be categorized into four classes: categorical, continuous, intensity, and event distribution measures.

Categorical measures assume values from a discrete, nonordered set of possibilities. Examples of categorical measures include network source and destination addresses, commands (e.g., commands that control data transfer and manage network connections), protocols, error codes (e.g., privilege violations, malformed service requests, and malformed packet codes), and port identifiers. The profiler engine 22 can build empirical distributions of the category values encountered, even if the list of possible values is open-ended. The engine 22 can have mechanisms for "aging out" categories whose long-term probabilities drop below a threshold.

Continuous measures assume values from a continuous or ordinal set. Examples include inter-event time (e.g., difference in time stamps between consecutive events from the same stream), counting measures such as the number of errors of a particular type observed in the recent past, the volume of data transfers over a period of time, and network traffic measures (number of packets and number of kilobytes). The profiler engine 22 treats continuous measures by first allocating bins appropriate to the range of values of the underlying measure, and then tracking the frequency of observation of each value range. In this way, multi-modal distributions are accommodated and much of the computational machinery used for categorical measures is shared. Continuous measures are useful not only for intrusion detection, but also to support the monitoring of the health and status of the network from the perspective of connectivity and throughput. For example, a measure of traffic volume maintained can detect an abnormal loss in the data rate of received packets when this volume falls outside historical norms. This sudden drop can be specific both to the network entity being monitored and to the time of day (e.g., the average sustained traffic rate for a major network artery is much different at 11:00 a.m. than at midnight).

Intensity measures reflect the intensity of the event stream (e.g., number of ICMP packets) over specified time intervals (e.g., 1 minute, 10 minutes, and 1 hour). Intensity measures are particularly suited for detecting flooding attacks, while also providing insight into other anomalies.

Event distribution measures are meta-measures that describes how other measures in the profile are affected by each event. For example, an "ls" command in an FTP session affects the directory measure, but does not affect measures related to file transfer. This measure is not interesting for all event streams. For example, all network-traffic event records affect the same measures (number of packets and kilobytes) defined for that event stream, so the event distribution does not change. On the other hand, event distribution measures are useful in correlative analysis performed by a monitor 16*a*–16*f* that receives reports from other monitors 16*a*–16*f*.

The system maintains and updates a description of behavior with respect to these measure types in an updated profile. The profile is subdivided into short-term and long-term profiles. The short-term profile accumulates values between updates, and exponentially ages (e.g., weighs data based on how long ago the data was collected) values for comparison to the long-term profile. As a consequence of the aging mechanism, the short-term profile characterizes recent activity, where "recent" is determined by a dynamically configurable aging parameters. At update time (typically, a time of low system activity), the update function folds the short-term values observed since the last update into the long-term profile, and the short-term profile is cleared. The long-term profile is itself slowly aged to adapt to changes in subject activity. Anomaly scoring compares related attributes in the short-term profile against the long-term profile. As all evaluations are done against empirical distributions, no assumptions of parametric distributions are made, and multi-modal and categorical distributions are accommodated. Furthermore, the algorithms require no a priori knowledge of intrusive or exceptional activity.

The statistical algorithm adjusts a short-term profile for the measure values observed in the event record. The distribution of recently observed values is compared against the long-term profile, and a distance between the two is obtained. The difference is compared to a historically adaptive deviation. The empirical distribution of this deviation is transformed to obtain a score for the event. Anomalous events are those whose scores exceed a historically adaptive score threshold based on the empirical score distribution.

This nonparametric approach handles all measure types and makes no assumptions on the modality of the distribution for continuous measures.

Profiles are provided to the computational engine as classes defined in the resource object 32. The mathematical functions for anomaly scoring, profile maintenance, and updating do not require knowledge of the data being analyzed beyond what is encoded in the profile class. Event collection interoperability supports translation of the event stream to the profile and measure classes. At that point, analysis for different types of monitored entities is mathematically similar. This approach imparts great flexibility to the analysis in that fading memory constants, update frequency, measure type, and so on are tailored to the network entity being monitored.

The measure types described above can be used individually or in combination to detect network packet attributes characteristic of intrusion. Such characteristics include large data transfers (e.g., moving or downloading files), an increase in errors (e.g., an increase in privilege violations or network packet rejections), network connection activity, and abnormal changes in network volume.

As shown, the monitor **16** also includes a signature engine **24**. The signature engine **24** maps an event stream against abstract representations of event sequences that are known to indicate undesirable activity. Signature-analysis objectives depend on which layer in the hierarchical analysis scheme the signature engine operates. Service monitor **16***a*–**16***c* signature engines **24** attempt to monitor for attempts to penetrate or interfere with the domain's operation. The signature engine scans the event stream for events that represent attempted exploitations of known attacks against the service, or other activity that stands alone as warranting a response from the monitor. Above the service layer, signature engines **24** scan the aggregate of intrusion reports from service monitors in an attempt to detect more global coordinated attack scenarios or scenarios that exploit interdependencies among network services. Layering signature engine analysis enables the engines **24** to avoid misguided searches along incorrect signature paths in addition to distributing the signature analysis.

A signature engines **24** can detect, for example, address spoofing, tunneling, source routing, SATAN attacks, and abuse of ICMP messages ("Redirect" and "Destination Unreachable" messages in particular). Threshold analysis is a rudimentary, inexpensive signature analysis technique that records the occurrence of specific events and, as the name implies, detects when the number of occurrences of that event surpasses a reasonable count. For example, monitors can encode thresholds to monitor activity such as the number of fingers, pings, or failed login requests to accounts such as guest, demo, visitor, anonymous FTP, or employees who have departed the company.

Signature engine **24** can also examine the data portion of packets in search of a variety of transactions that indicate suspicious, if not malicious, intentions by an external client. The signature engine **24**, for example, can parse FTP traffic traveling through the firewall or router for unwanted transfers of configuration or specific system data, or anonymous requests to access non-public portions of the directory structure. Similarly, a monitor can analyze anonymous FTP sessions to ensure that the file retrievals and uploads/modifications are limited to specific directories. Additionally, signature analysis capability can extend to session analyses of complex and dangerous, but highly useful, services like HTTP or Gopher.

Signature analysis can also scan traffic directed at unused ports (i.e., ports to which the administrator has not assigned a network service). Here, packet parsing can be used to study network traffic after some threshold volume of traffic, directed at an unused port, has been exceeded. A signature engine **24** can also employ a knowledge base of known telltale packets that are indicative of well-known network-service protocol traffic (e.g., FTP, Telnet, SMTP, HTTP). The signature engine **24** then determines whether the unknown port traffic matches any known packet sets. Such comparisons could lead to the discovery of network services that have been installed without an administrator's knowledge.

The analysis engines **22**, **24** receive large volumes of events and produce smaller volumes of intrusion or suspicion reports that are then fed to the resolver **20**. The resolver **20** is an expert system that receives the intrusion and suspicion reports produced by the analysis engines **22**, **24** and reports produced externally by other analysis engines to which it subscribes. Based on these reports, the resolver **20** invokes responses. Because the volume of intrusion and suspicion reports is lower than the volume of events received by the analysis engines **22**, **24**, the resolver **20** can afford the more sophisticated demands of configuration maintenance and managing the response handling and external interfaces necessary for monitor operation. Furthermore, the resolver **20** adds to extensibility by providing the subscription interface through which third-party analysis tools **28**, **30** can interact and participate in the hierarchical analysis scheme.

Upon its initialization, the resolver **20** initiates authentication and subscription sessions with those monitors **16***a*–**16***f* whose identities appear in the monitor's **16** subscription-list (**46** FIG. 3). The resolver **20** also handles all incoming requests by subscribers, which must authenticate themselves to the resolver **20**. Once a subscription session is established with a subscriber monitor, the resolver **20** acts as the primary interface through which configuration requests are received and intrusion reports are disseminated.

Thus, resolvers **20** can request and receive reports from other resolvers at lower layers in the analysis hierarchy. The resolver **20** forwards analysis reports received from subscribees to the analysis engines **22**, **24**. This tiered collection and correlation of analysis results allows monitors **16***a*–**16***f* to represent and profile global malicious or anomalous activity that is not visible locally.

In addition to external-interface responsibilities, the resolver **20** operates as a fully functional decision engine, capable of invoking real-time response measures in response to malicious or anomalous activity reports produced by the analysis engines. The resolver **20** also operates as the center of intramonitor communication. As the analysis engines **22**, **24** build intrusion and suspicion reports, they propagate these reports to the resolver **20** for further correlation, response, and dissemination to other monitors **16***a*–**16***f*. The resolver **20** can also submit runtime configuration requests to the analysis engines **22**, **24**, for example, to increase or decrease the scope of analyses (e.g., enable or disable additional signature rules) based on various operating metrics. These configuration requests could be made as a result of encountering other intrusion reports from other subscribers. For example, a report produced by a service monitor **16***a*–**16***c* in one domain could be propagated to an enterprise monitor **16***f*, which in turn sensitizes service monitors in other domains to the same activity.

The resolver **20** also operates as the interface mechanism between administrators and the monitor **16**. From the per-

spective of a resolver 20, the administrator interface is simply a subscribing service to which the resolver 20 may submit reports and receive configuration requests. An administrative interface tool can dynamically subscribe and unsubscribe to any of the deployed resolvers 20, as well as submit configuration requests and asynchronous probes as desired.

The monitors 16*a*–16*f* incorporate a bidirectional messaging system that uses a standard interface specification for communication within and between monitor elements and external modules. Using this interface specification, third-party modules 28, 30 can communicate with monitors. For example, third-party modules 28 can submit event records to the analysis engines 22, 24 for processing. Additionally, third-party modules 30 may also submit and receive analysis results via the resolver's 20 external interfaces. Thus, third-party modules 28, 30 can incorporate the results from monitors into other surveillance efforts or contribute their results to other monitors 16*a*–16*f*. Lastly, the monitor's 16 internal API allows third-party analysis engines to be linked directly into the monitor boundary.

The message system operates under an asynchronous communication model for handling results dissemination and processing that is generically referred to as subscription-based message passing. Component interoperation is client/server-based, where a client module may subscribe to receive event data or analysis results from servers. Once a subscription request is accepted by the server, the server module forwards events or analysis results to the client automatically as data becomes available, and may dynamically reconfigure itself as requested by the client's control requests. This asynchronous model reduces the need for client probes and acknowledgments.

The interface supports an implementation-neutral communication framework that separates the programmer's interface specification and the issues of message transport. The interface specification embodies no assumptions about implementation languages, host platform, or a network. The transport layer is architecturally isolated from the internals of the monitors so that transport modules may be readily introduced and replaced as protocols and security requirements are negotiated between module developers. The interface specification involves the definition of the messages that the various intrusion-detection modules must convey to one another and how these messages should be processed. The message structure and content are specified in a completely implementation-neutral context.

Both intramonitor and intermonitor communication employ identical subscription-based client-server models. With respect to intermonitor communication, the resolver 20 operates as a client to the analysis engines, and the analysis engines 22, 24 operate as clients to the event filters. Through the internal message system, the resolver 20 submits configuration requests to the analysis engines 22, 24, and receives from the analysis engines 22, 24 their analysis results. The analysis engines 22, 24 operate as servers providing the resolver 20 with intrusion or suspicion reports either asynchronously or upon request. Similarly, the analysis engines 22, 24 are responsible for establishing and maintaining a communication link with an event collection method (or event filter) and prompting the reconfiguration of the collection method's filtering semantics when necessary.

Intermonitor communication also operates using the subscription-based hierarchy. A domain monitor 16*d*–16*e* subscribes to the analysis results produced by service monitors 16*a*–16*c*, and then propagates its own analytical reports to its parent enterprise monitor 16*f*. The enterprise monitor 16*f* operates as a client to one or more domain monitors 16*d*–16*e*, allowing them to correlate and model enterprise-wide activity from the domain-layer results. Domain monitors 16*d*–16*e* operate as servers to the enterprise monitors 16*f*, and as clients to the service monitors 16*a*–16*c* deployed throughout their domain 12*a*–12*c*. This message scheme can operate substantially the same if correlation were to continue at higher layers of abstraction beyond enterprise 10 analysis.

Intramonitor and intermonitor programming interfaces are substantially the same. These interfaces can be subdivided into five categories of interoperation: channel initialization and termination, channel synchronization, dynamic configuration, server probing, and report/event dissemination. Clients are responsible for initiating and terminating channel sessions with servers. Clients are also responsible for managing channel synchronization in the event of errors in message sequencing or periods of failed or slow response (i.e., "I'm alive" confirmations). Clients may also submit dynamic configuration requests to servers. For example, an analysis engine 22, 24 may request an event collection method to modify its filtering semantics. Clients may also probe servers for report summaries or additional event information. Lastly, servers may send clients intrusion/suspicion reports in response to client probes or in an asynchronous dissemination mode.

The second part of the message system framework involves specification of a transport mechanism used to establish a given communication channel between monitors 16*a*–16*f* or possibly between a monitor 16*a*–16*f* and a third-party security module. All implementation dependencies within the message system framework are addressed by pluggable transport modules. Transport modules are specific to the participating intrusion-detection modules, their respective hosts, and potentially to the network—should the modules require cross-platform interoperation. Instantiating a monitor 16*a*–16*f* may involve incorporation of the necessary transport module(s) (for both internal and external communication).

The transport modules that handle intramonitor communication may be different from the transport modules that handle intermonitor communication. This allows the intramonitor transport modules to address security and reliability issues differently than how the intermonitor transport modules address security and reliability. While intramonitor communication may more commonly involve interprocess communication within a single host, intermonitor communication will most commonly involve cross-platform networked interoperation. For example, the intramonitor transport mechanisms may employ unnamed pipes which provides a kernel-enforced private interprocess communication channel between the monitor 16 components (this assumes a process hierarchy within the monitor 16 architecture). The monitor's 16 external transport, however, will more likely export data through untrusted network connections and thus require more extensive security management. To ensure the security and integrity of the message exchange, the external transport may employ public/private key authentication protocols and session key exchange. Using this same interface, third-party analysis tools may authenticate and exchange analysis results and configuration information in a well-defined, secure manner.

The pluggable transport permits flexibility in negotiating security features and protocol usage with third parties. Incorporation of a commercially available network management system can deliver monitoring results relating to security, reliability, availability, performance, and other

attributes. The network management system may in turn subscribe to monitor produced results in order to influence network reconfiguration.

All monitors (service, domain, and enterprise) 16*a*–16*f* use the same monitor code-base. However, monitors may include different resource objects 32 having different configuration data and methods. This reusable software architecture can reduce implementation and maintenance efforts. Customizing and dynamically configuring a monitor 16 thus becomes a question of building and/or modifying the resource object 32.

Referring to FIG. 3, the resource object 32 contains the operating parameters for each of the monitor's 16 components as well as the analysis semantics (e.g., the profiler engine's 22 measure and category definition, or the signature engine's 24 penetration rule-base) necessary to process an event stream. After defining a resource object 32 to implement a particular set of analyses on an event stream, the resource object 32 may be reused by other monitors 16 deployed to analyze equivalent event streams. For example, the resource object 32 for a domain's router may be reused as other monitors 16 are deployed for other routers in a domain 12*a*–12*c*. A library of resource objects 32 provides prefabricated resource objects 32 for commonly available network entities.

The resource object 32 provides a pluggable configuration module for tuning the generic monitor code-base to a specific event stream. The resource object 32 includes configurable event structures 34, analysis unit configuration 38*a*–38*n*, engine configuration 40*a*–40*n*, resolver configuration 42, decision unit configuration 44, subscription list data 46, and response methods 48.

Configurable event structures 34 define the structure of event records and analysis result records. The monitor code-base maintains no internal dependence on the content or format of any given event stream or the analysis results produced from analyzing the event stream. Rather, the resource object 32 provides a universally applicable syntax for specifying the structure of event records and analysis results. Event records are defined based on the contents of an event stream(s). Analysis result structures are used to package the findings produced by analysis engines. Event records and analysis results are defined similarly to allow the eventual hierarchical processing of analysis results as event records by subscriber monitors.

Event-collection methods 36 gather and parse event records for analysis engine processing. Processing by analysis engines is controlled by engine configuration 40*a*–40*n* variables and data structures that specify the operating configuration of a fielded monitor's analysis engine(s). The resource object 32 maintains a separate collection of operating parameters for each analysis engine instantiated in the monitor 16. Analysis unit configuration 38*a*–38*n* include configuration variables that define the semantics employed by the analysis engine to process the event stream.

The resolver configuration 42 includes operating parameters that specify the configuration of the resolver's internal modules. The decision unit configuration 44 describes semantics used by the resolver's decision unit for merging the analysis results from the various analysis engines. The semantics include the response criteria used to invoke countermeasure handlers. A resource object 32 may also include response methods 48. Response methods 48 include preprogrammed countermeasure methods that the resolver may invoke as event records are received. A response method 48 includes evaluation metrics for determining the circumstances under which the method should be invoked. These metrics include a threshold metric that corresponds to the measure values and scores produced by the profiler engine 22 and severity metrics that correspond to subsets of the associated attack sequences defined within the resource object 32.

Countermeasures range from very passive responses, such as report dissemination to other monitors 16*a*–16*f* or administrators, to highly aggressive actions, such as severing a communication channel or the reconfiguration of logging facilities within network components (e.g., routers, firewalls, network services, audit daemons). An active response may invoke handlers that validate the integrity of network services or other assets to ensure that privileged network services have not been subverted. Monitors 16*a*–16*f* may invoke probes in an attempt to gather as much counterintelligence about the source of suspicious traffic by using features such as traceroute or finger.

The resource object 32 may include a subscription list 46 that includes information necessary for establishing subscription-based communication sessions, which may include network address information and public keys used by the monitor to authenticate potential clients and servers. The subscription list 46 enables transmission or reception of messages that report malicious or anomalous activity between monitors. The most obvious examples where relationships are important involve interdependencies among network services that make local policy decisions. For example, the interdependencies between access checks performed during network file system mounting and the IP mapping of the DNS service. An unexpected mount monitored by the network file system service may be responded to differently if the DNS monitor informs the network file system monitor of suspicious updates to the mount requestor's DNS mapping.

The contents of the resource object 32 are defined and utilized during monitor 16 initialization. In addition, these fields may be modified by internal monitor 16 components, and by authorized external clients using the monitor's 16 API. Modifying the resource object 32 permits adaptive analysis of an event stream, however, it also introduces a potential stability problem if dynamic modifications are not tightly restricted to avoid cyclic modifications. To address this issue, monitors 16 can be configured to accept configuration requests from only higher-level monitors 16.

Referring to FIG. 4, a monitor performs network surveillance by monitoring 66 a stream of network packets. The monitor builds a statistical model of network activity from the network packets, for example, by building 68 long-term and short-term statistical profiles from measures derived from the network packets. The measures include measures that can show anomalous network activity characteristic of network intrusion such as measures that describe data transfers, network connections, privilege and network errors, and abnormal levels of network traffic. The monitor can compare 70 the long-term and short-term profiles to detect suspicious network activity. Based on this comparison, the monitor can respond 72 by reporting the activity to another monitor or by executing a countermeasure response. More information can be found in P. Porras and A. Valdes "Live Traffic Analysis of TCP/IP Gateways", Networks and Distributed Systems Security Symposium, March 1998, which is incorporated by reference in its entirety.

A few examples can illustrate this method of network surveillance. Network intrusion frequently causes large data

transfers, for example, when an intruder seeks to download sensitive files or replace system files with harmful substitutes. A statistical profile to detect anomalous data transfers might include a continuous measure of file transfer size, a categorical measure of the source or destination directory of the data transfer, and an intensity measure of commands corresponding to data transfers (e.g., commands that download data). These measures can detect a wide variety of data transfer techniques such as a large volume of small data transfers via e-mail or downloading large files en masse. The monitor may distinguish between network packets based on the time such packets were received by the network entity, permitting statistical analysis to distinguish between a normal data transfer during a workday and an abnormal data transfer on a weekend evening.

Attempted network intrusion may also produce anomalous levels of errors. For example, categorical and intensity measures derived from privilege errors may indicate attempts to access protected files, directories, or other network assets. Of course, privilege errors occur during normal network operation as users mistype commands or attempt to perform an operation unknowingly prohibited. By comparing the long-term and short-term statistical profiles, a monitor can distinguish between normal error levels and levels indicative of intrusion without burdening a network administrator with the task of arbitrarily setting an unvarying threshold. Other measures based on errors, such as codes describing why a network entity rejected a network packet enable a monitor to detect attempts to infiltrate a network with suspicious packets.

Attempted network intrusion can also be detected by measures derived from network connection information. For example, a measure may be formed from the correlation (e.g., a ratio or a difference) of the number of SYN connection request messages with the number of SYN_ACK connection acknowledgment messages and/or the number of ICMP messages sent. Generally, SYN requests received should balance with respect to the total of SYN_ACK and ICMP messages sent. That is, flow into and out-of a network entity should be conserved. An imbalance can indicate repeated unsuccessful attempts to connect with a system, perhaps corresponding to a methodical search for an entry point to a system. Alternatively, intensity measures of transport-layer connection requests, such as a volume analysis of SYN-RST messages, could indicate the occurrence of a SYN-attack against port availability or possibly port-scanning. Variants of this can include intensity measures of TCP/FIN messages, considered a more stealthy form of port scanning.

Many other measures can detect network intrusion. For example, "doorknob rattling," testing a variety of potentially valid commands to gain access (e.g., trying to access a "system" account with a password of "system"), can be detected by a variety of categorical measures. A categorical measure of commands included in network packets can identify an unusual short-term set of commands indicative of "doorknob-rattling." Similarly, a categorical measure of protocol requests may also detect an unlikely mix of such requests.

Measures of network packet volume can also help detect malicious traffic, such as traffic intended to cause service denials or perform intelligence gathering, where such traffic may not necessarily be violating filtering policies. A measure reflecting a sharp increase in the overall volume of discarded packets as well as a measure analyzing the disposition of the discarded packets can provide insight into unintentionally malformed packets resulting from poor line quality or internal errors in neighboring hosts. High volumes of discarded packets can also indicate more maliciously intended transmissions such as scanning of UPD ports or IP address scanning via ICMP echoes. Excessive number of mail expansion request commands (EXPN) may indicate intelligence gathering, for example, by spammers.

A long-term and short-term statistical profile can be generated for each event stream. Thus, different event streams can "slice" network packet data in different ways. For example, an event stream may select only network packets having a source address corresponding to a satellite office. Thus, a long-term and short-term profile will be generated for the particular satellite office. Thus, although a satellite office may have more privileges and should be expected to use more system resources than other external addresses, a profile of satellite office use can detect "address spoofing" (i.e., modifying packet information to have a source address of the satellite office).

The same network packet event may produce records in more than one event stream. For example, one event stream may monitor packets for FTP commands while another event stream monitors packets from a particular address. In this case, an FTP command from the address would produce an event record in each stream.

Referring to FIG. 5, a monitor may also "deinterleave." That is, the monitor may create and update **74**, **76** more than one short-term profile for comparison **78** against a single long-term profile by identifying one of the multiple short-term profiles that will be updated by an event record in an event stream. For example, at any one time a network entity may handle several FTP "anonymous" sessions. If each network packet for all anonymous sessions were placed in a single short-term statistical profile, potentially intrusive activity of one anonymous session may be statistically ameliorated by non-intrusive sessions. By creating and updating short-term statistical profiles for each anonymous session, each anonymous session can be compared against the long-term profile of a normal FTP anonymous session. Deinterleaving can be done for a variety of sessions including HTTP sessions (e.g., a short-term profile for each browser session).

Referring to FIG. 6, a computer platform **14** suitable for executing a network monitor **16** includes a display **50**, a keyboard **54**, a pointing device **58** such as a mouse, and a digital computer **56**. The digital computer **56** includes memory **62**, a processor **60**, a mass storage device **64***a*, and other customary components such as a memory bus and peripheral bus. The platform **14** may further include a network connection **52**.

Mass storage device **64***a* can store instructions that form a monitor **16**. The instructions may be transferred to memory **62** and processor **60** in the course of operation. The instructions **16** can cause the display **50** to display images via an interface such as a graphical user interface. Of course, instructions may be stored on a variety of mass storage devices such as a floppy disk **64***b*, CD-ROM **64***c*, or PROM (not shown).

Other embodiments are within the scope of the following claims.

What is claimed is:

1. A method of network surveillance, comprising:

receiving network packets handled by a network entity;

building at least one long-term and at least one short-term statistical profile from at least one measure of the network packets, the at least one measure monitoring data transfers, errors, or network connections;

comparing at least one long-term and at least one short-term statistical profile; and

determining whether the difference between the short-term statistical profile and the long-term statistical profile indicates suspicious network activity.

2. The method of claim 1, wherein the measure monitors data transfers by monitoring network packet data transfer commands.

3. The method of claim 1, wherein the measure monitors data transfers by monitoring network packet data transfer errors.

4. The method of claim 1, wherein the measure monitors data transfers by monitoring network packet data transfer volume.

5. The method of claim 1, wherein the measure monitors network connections by monitoring network connection requests.

6. The method of claim 1, wherein the measure monitors network connections by monitoring network connection denials.

7. The method of claim 1, wherein the measure monitors network connections by monitoring a correlation of network connections requests and network connection denials.

8. The method of claim 1, wherein the measure monitors errors by monitoring error codes included in a network packet.

9. The method of claim 8, wherein an error code comprises a privilege error code.

10. The method of claim 8, wherein an error code comprises an error code indicating a reason a packet was rejected.

11. The method of claim 1, further comprising responding based on the determining whether the difference between the short-term statistical profile and the long-term statistical profile indicates suspicious network activity.

12. The method of claim 11, wherein responding comprises transmitting an event record to a network monitor.

13. The method of claim 12, wherein transmitting the event record to a network monitor comprises transmitting the event record to a hierarchically higher network monitor.

14. The method of claim 13, wherein transmitting the event record to a network monitor comprises transmitting the event record to a network monitor that receives event records from multiple network monitors.

15. The method of claim 14, wherein the monitor that receives event records from multiple network monitors comprises a network monitor that correlates activity in the multiple network monitors based on the received event records.

16. The method of claim 11, wherein responding comprises altering analysis of the network packets.

17. The method of claim 11, wherein responding comprises severing a communication channel.

18. The method of claim 1, wherein the network packets comprise TCP/IP packets.

19. The method of claim 1, wherein the network entity comprises a gateway, a router, or a proxy server.

20. The method of claim 1, wherein the network entity comprises a virtual private network entity.

21. A method of network surveillance, comprising:

monitoring network packets handled by a network entity;

building a long-term and multiple short-term statistical profiles of the network packets;

comparing one of the multiple short-term statistical profiles with the long-term statistical profile; and

determining whether the difference between the one of the multiple short-term statistical profiles and the long-term statistical profile indicates suspicious network activity.

22. The method of claim 21, wherein the multiple short-term statistical profiles comprise profiles that monitor different anonymous FTP sessions.

23. The method of claim 21, wherein building multiple short-term statistical profiles comprises deinterleaving packets to identify a short-term statistical profile.

24. A computer program product, disposed on a computer readable medium, the product including instructions for causing a processor to:

receive network packets handled by a network entity;

build at least one long-term and at least one short-term statistical profile from at least one measure of the network packets, the measure monitoring data transfers, errors, or network connections;

compare at least one short-term and at least one long-term statistical profile; and

determine whether the difference between the short-term statistical profile and the long-term statistical profile indicates suspicious network activity.

25. A method of network surveillance, comprising:

receiving packets at a virtual private network entity; and

building at least one long-term and at least one short-term statistical profile based on the received packets, and

comparing at least one long-term statistical profile with at least one short-term statistical profile to determine whether the packets indicate suspicious network activity.

26. The method of claim 25, further comprising decrypting the packets before statistically analyzing the packets.

27. The method of claim 25, further comprising not decrypting the packets before statistically analyzing the packets.

* * * * *

J

Technical Report
(CDRL A005)

Architecture Design of a Scalable Intrusion Detection System for the Emerging Network Infrastructure

DARPA Order Number: E296

Issued by Rome Lab under
Contract Number: F30602-96-C0325

Submitted: April 1997

by

Y. Frank Jou
Fengmin Gong
Chandru Sargor

MCNC
Information Technologies Division
Research Triangle Park, N.C. 27709

Shyhtsun Felix Wu
W. Rance Cleaveland

Dept. of Computer Science
North Carolina State University
Raleigh, N.C. 27695-7914

Contents

1 Introduction 1
- 1.1 Background . . . 1
- 1.2 Organization . . . 2

2 Intrusion Detection System Architecture 3
- 2.1 Architecture Overview . . . 3
- 2.2 Components . . . 3
  - 2.2.1 Local Subsystem . . . 3
    - 2.2.1.1 Interception/Redirection Module . . . 5
    - 2.2.1.2 Prevention Module . . . 5
    - 2.2.1.3 Detection Module . . . 5
      - 2.2.1.3.1 Statistical Analysis Module . . . 5
      - 2.2.1.3.2 Protocol Analysis Module . . . 5
    - 2.2.1.4 Local Decision Module . . . 6
    - 2.2.1.5 Information Abstraction Module (IAM) . . . 6
    - 2.2.1.6 JiNao Management Information Base (JiNaoMIB) . . . 6
  - 2.2.2 Remote Subsystem . . . 6
  - 2.2.3 Management Information Exchange Protocol . . . 7
- 2.3 Interfaces Between Modules . . . 7

3 Design Objectives 12
- 3.1 Comprehensiveness . . . 13
- 3.2 Scalability . . . 13
- 3.3 Interoperability . . . 13

4 Functional Description 14
- 4.1 Local Subsystem . . . 14
  - 4.1.1 Interception/Redirection Module . . . 14
  - 4.1.2 Prevention Module . . . 16
    - 4.1.2.1 Prevention Layer . . . 16
    - 4.1.2.2 Extraction Layer . . . 16
  - 4.1.3 Detection Module . . . 18
    - 4.1.3.1 Statistical Analysis Module . . . 18
      - 4.1.3.1.1 Components of the Statistical Approach . . . 19
      - 4.1.3.1.2 Computing the Q statistics for the intensity measures . . . 20
      - 4.1.3.1.3 Computing the Q statistics for the audit record distribution measures . . . 21
      - 4.1.3.1.4 Computing the frequency distribution for Q . . . 22
      - 4.1.3.1.5 Deriving S from Q for the intensity measures . . . 22
      - 4.1.3.1.6 Deriving S from Q for the audit record distribution measure . . . 23
      - 4.1.3.1.7 Training and updating . . . 24

4.1.3.2 Protocol Analysis Module 24
4.1.3.2.1 Overview 24
4.1.3.2.2 Interface 24
4.1.3.2.3 Implementation 25
4.1.4 Local Decision Module (LDecM) 29
4.1.4.1 Functional Description 29
4.1.4.2 Examples 30
4.1.4.3 Exceptions and Errors 31
4.1.4.4 Remarks 31
4.1.5 Information Abstraction Module (IAM) 31
4.1.5.1 IAM Functions 31
4.1.5.1.1 Local detection information aggregation and MIB-fication 31
4.1.5.1.2 Periodic checking and propagation of global information 32
4.1.5.2 Interface Mechanisms and Formats 32
4.1.5.3 Scope of Impact Representation 32
4.1.6 Management Information Base (MIB) 32
4.2 Management Information Exchange Protocol 34
4.3 Remote JiNao Management Applications 35

List of Tables

List of Figures

1 Ji-Nao System Architecture. 4
2 Packet formats across protocol layers. 15
3 Interception and prevention modules and their relationship. 17
4 Figurative description of catch and trap. 35

Design of a Scalable Intrusion Detection System for the Emerging Network Infrastructure

1 Introduction

This document describes the design of a scalable intrusion detection system funded by DARPA through Contract No. F30602-96-C0325. This three-year project aims at designing and developing a software system for protecting against intruders from breaking into network routers, switches, and network management channels. The project is a joint collaboration between MCNC and North Carolina State University (NCSU).

Given the increasing popularity of the Internet, intrusion incidents are becoming common events of life. Some of these incidents are simply out of innocuous curiosity. Some, however, are due to malicious attempts in order to compromise the availability of information system or the integrity and privacy of the information itself. Despite the best efforts of the protocol designers, implementors, and system administrators, it is prudent to assume that attacks will occur and some, unfortunately, will succeed. Therefore, it is vitally important to develop means to automatically detect and respond to these attacks in order to maintain critical information services.

Depending on the goals of the intruder, the targets of attack may range from individual end hosts to a network of routers and switches. In this project, we focus our effort on the protection of the network infrastructure since the attacks on the routers/switches have the potential of disrupting a large scale of information services on which the national defense and economy may depend. Our goal of designing this detection system is to provide a comprehensive approach which leverages on the application of novel detection techniques together with extension of some existing host-based intrusion detection methods in an internetworking environment. In particular, we will conduct logical and statistical analysis of network routing and management protocols to construct a scalable distributed intrusion detection system for the emerging internetwork environment.

1.1 Background

Intrusive activity is occurring on our computer systems. Reports frequently appear in the media about outsiders breaking into computers, employees misusing computers, and rogue viruses and worms penetrating computer systems. Due to these incidents, we have seen a growing interest in computer system intrusion detection in the last several years. The earliest work in this area was a study by Jim Anderson [1]. Anderson categorized the threats as:

- Masquerader: An individual who is not authorized to use the computer, and who penetrates a system's access controls to exploit a legitimate user's account.
- Misfeasor: A legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuses his or her privileges.

- Clandestine users: An individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection altogether.

Anderson suggested that masqueraders can be detected either by auditing failed login attempts or by observing departures from established patterns of use for individual users. Misfeasors can be detected by observing failed access attempts to files, programs, and other resources. His suggestion for detecting the clandestine user is to monitor certain system-wide parameters, such as CPU, memory, and disk activity, and compare these with what has been historically established as "usual" or "normal" for that facility. All of these approaches have been adopted one way or the other by subsequent studies.

Dorothy Denning [4] and her colleagues at SRI International undertook a project for developing an intrusion detection expert system (IDES) prototype. Denning proposed to monitor standard operations on a target system for deviations in usage. Her early research tried to define the activities and statistical measures best suited to do this detection. Teresa Lunt [5] and her colleagues continue this research with the development of the IDES system. They expanded the original concept by adding an expert system component that addresses known or suspected security flaws in the target system. IDES (and its follow-up Next generation IDES, or NIDES) system research has served to demonstrate two things. First, statistical analysis of computer system activities provides a characterization of normal system and user behavior, and activities deviating beyond normal bounds is detectable. Second, known intrusion scenarios, exploitation of known system vulnerabilities, and violations of a system's security policy are detectable through use of a rule-based expert system.

In the early stage, intrusion detection system were designed around the analysis of a single host's audit trail. With the proliferation of computer networks, many of the intrusion detection systems began to extend the techniques to networks of computers. Most of the current network intrusion detection efforts have taken one of the two following approaches. One approach is to collect data from separate hosts on a network for processing by a centralized intrusion detection system [2][3]. The other approach is to target network traffic at the service and protocol levels [6][7]. Our effort is close to the second approach with a few exceptions. First, we are interested in protecting network infrastructure and particularly focus on routing and management capabilities. Therefore, the target of analysis is mainly on specific protocol traffic instead of general data traffic. Second, the proposed protocol analysis approach in our architecture design is unique which analyzes the logical behavior of routing and management protocols in order to identify the set of states that are indicative of security attacks. Third, network management functionalities are part of the integrated system design. Through these functionalities, the intrusion detection system can be incorporated into existing management framework as an extension of fault management.

1.2 Organization

Section 2 provides an overview of the architecture of a model intrusion detection system and introduce its components and associated functional requirements. Section 3 outlines the design objectives and system features for the experimental system being implemented by the authors. Finally, detailed description of a functional overview is given in Section 4.

2 Intrusion Detection System Architecture

In this section we present an overview of the system architecture design. The system consists of complementary functional blocks for providing comprehensive detection capabilities. It also incorporates standard network management functionalities to lay a foundation for facilitating automated responses in future research efforts. A brief description of each system component and its functional requirements will be given later in the section.

2.1 Architecture Overview

Figure 1 illustrates the architecture design of our intrusion detection system. At the top level, there are two subsystems: namely, local detection subsystem and remote management subsystem. The remote management unit implements a set of network management applications which can both probe the status of and issue commands to the local detection subsystem. It is one of our design objectives that the management applications will be based on SNMP such that the management function can be easily incorporated into any existing SNMP based network management platform.

A local subsystem is associated with a router/switch to function as a security filter and analyze the incoming packets from its neighbors. The transaction record with each neighbor will be maintained separately. If any of its neighbor routers/switches behaves differently from its historical norm or transitions into an improper protocol state, then it may be an indication that this neighbor is either faulty or compromised. Depending on the degree of deviation or the nature of fault/attack, an alert or alarm signal will be issued to acquire the security officer's attention.

A remote management subsystem can oversee several routers/switches. Some intrusions, like doorknob rattling attack, which may be difficult to detect at a local level can be made easier by checking the global status across several routers/switches. While it is not within the scope of this project, we expect that the detection/analysis functions implemented in the local subsystem can be extended to a global level and correlate intrusion events among several routers. The management capability, which is based on SNMP framework, can logically be further extended among management nodes in a hierarchical fashion to establish a status map for an autonomous system.

2.2 Components

The functional requirements of the components shown in Figure 1 are described in the following sections.

2.2.1 Local Subsystem

A local subsystem consists of the following modules: interception/redirection module, rule-based prevention module, protocol and statistical-based detection modules, decision module, and information abstraction module. It also includes a management information base (MIB) and remote MIB agent functions which provide access to remote management applications. A brief description for each module follows.

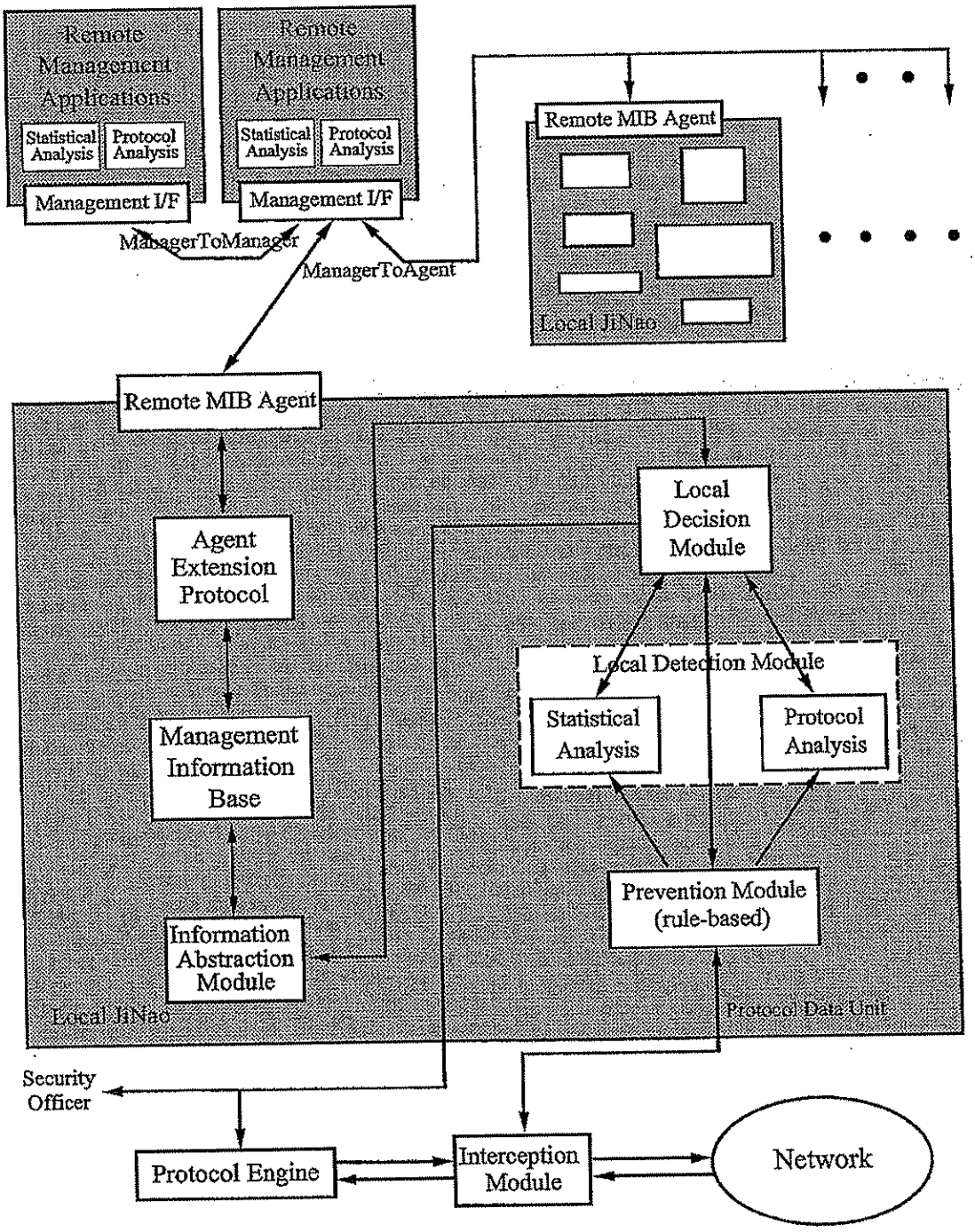


Figure 1: Ji-Nao System Architecture.

2.2.1.1 Interception/Redirection Module The responsibility of the interception module is to *redirect* the target protocol information flow to the prevention layer for rule checking. If possible, the redirected information flow should be timestamped. After receiving a clear signal from the prevention module, the interception module will release the packet to the protocol engine for execution. The interception module can also facilitate the capability of active intrusion detection (catch-and-trap, explained in Section 4.3) by holding up an outgoing packet temporarily.

2.2.1.2 Prevention Module As the name "prevention" implies, this module will implement a small set of administrative rules to filter out any packet with clear security violation before it enters into the router. The intent of the design is for this module to serve as a gate-keeper with a very short response time. The packets to be discarded include all those that may have significant damaging effect on the infrastructure according to general security guidelines or special security concerns of an administrative domain. The mechanism of this module is similar to a small rule-based expert system found in a conventional intrusion detection system.

2.2.1.3 Detection Module If a packet passes through the prevention module, it will be forwarded to the protocol engine for execution and to the local detection module which performs both statistical- and protocol-based intrusion checks. The results of these checks are sent to the local decision module. As shown in the Figure 1, the architecture allows for a two way interaction between the prevention/detection modules and the decision module. Specifically, it allows for the set of checking rules and their associated parameters in both prevention and detection modules be dynamically modified by the decision information in response to the input of detection information. The decision information can be either derived from the input of detection modules or come from the global detection module and the remote management applications through the MIB interface. Therefore, our design will allow for a certain degree of automated responses through the adoption of network management framework.

2.2.1.3.1 Statistical Analysis Module Intrusion detection using statistical analysis is founded on the contention that behavioral signatures exist for either users' usage profiles or protocol execution patterns (in this case, network routing and management protocols) and intrusion will result in abnormal signatures. Any behavior deviating from the normal signature will be considered as an anomaly and appropriate alarms can be triggered. This module provides the capability to detect intrusions that exploit previously unknown vulnerabilities. It is intended to uncover those attacks that cannot be prevented by a set of rules embedded in a rule-based component or cannot be detected by security analysis conducted through protocol-based approach.

2.2.1.3.2 Protocol Analysis Module The protocol-based approach detects intrusion by monitoring the execution of protocols in a router and triggering intrusion alarms when an anomalous state is entered. Specifically, we will investigate two routing protocols

(OSPF and PNNI, the latter will be contingent upon the availability of public domain implementation of PNNI routing software) and one network management information exchange protocol (SNMP). OSPF is expected to eventually replace RIP as the primary choice of interior gateway protocol and PNNI is standardized as the routing protocol for private ATM networks. SNMP is also a standard-track Internet network management protocol. The standardization of SNMPv3 is currently work in progress at IETF. Some of the vulnerabilities of these protocols have been reported in the proposal. We continue this effort to identify potential security weaknesses and propose possible attack scenarios.

2.2.1.4 Local Decision Module The decision module on one hand serves as a coordinator to correlate the information from the prevention and the detection modules for determining if any intrusion has occurred and what actions need to be taken. On the other, it issues commands to either update and/or activate rules in prevention module, or modify parameters/options in local detection modules.

Through both local and remote MIB agents, the decision module provides its local view of neighbor status to remote management applications for identifying any global scale of attacks. It also relays commands from remote management applications to the local prevention module and detection modules.

2.2.1.5 Information Abstraction Module (IAM) IAM serves as an interface module between the JiNao local intrusion detection subsystem and the remote JiNao modules as well other network management applications. In propagating local intrusion detection results to the outside, the IAM aggregates local detection results and converts them into MIB format. In updating the local detection and prevention modules with new rule sets via the local decision module, the IAM receives and processes requests from the remote subsystems through the JiNao MIB interface.

2.2.1.6 JiNao Management Information Base (JiNaoMIB) As part of the network management framework, the JiNao management information base is a collection of MIB variables that are of interest to the remote management subsystems. JiNao MIB variables include detection results, decision information, rule and finite state machine configurations, log access and other security control options. Some of the MIB variables are readable and/or writable, and conceptually representing the interface for several security control and management operations.

2.2.2 Remote Subsystem

In the current scope of the project, a remote subsystem consists of a set of management applications for monitoring and controlling a few local detection subsystems. It is expected that a management application would be able to re-configure the local detection system dynamically. With this configurability, the local detection subsystem can respond to intrusion differently under different situations.

Ideally, as a natural extension of the current scope of the project, we expect that a remote subsystem can also implement similar detection capabilities in order to detect a larger scale of orchestrated attack. We realize that some attacks (for instance, door-knob rattling attack)

can only be detected on a more global scale. One approach in dealing with these attacks is for the management applications to communicate with their local detection agents in order to form a global view of the domain surrounding this remote subsystem. The notion of global detection can be further extended to cover more than one remote subsystem, either in a distributed or hierarchical fashion.

2.2.3 Management Information Exchange Protocol

The management information exchange protocol (*e.g.,* SNMP) provides communication channels between remote management applications and local MIB agents (manager to agent) or between any pair of remote subsystems (manager to manager). It allows the management applications to access local MIB variables, control the execution of protocol engines, and modify the operation of local detection system. A local MIB agent can issue a trap command to get a management application's attention when a special event occurs. Two or more remote subsystems can establish a global view of the network by exchanging detection information from their domains. If the remote subsystems are organized in a distributed fashion, the communication among them is through manager to manager operation. Otherwise, the communication will be manager to agent operation when the system is in a hierarchical architecture.

A MIB agent can be sub-divided into two different types:

Master agent: A single processing entity which sends and receives SNMP protocol messages in an agent role but typically has little or no direct access to management information (or MIB variables). Remote management applications talk SNMP directly to the master agent.

Subagent: Zero or more processing entities called subagents, which are shielded from the SNMP PDUs processed by the master agent, but which has access to management information. The master agent communicates via some agent extension protocols (*e.g.,* SMUX [8], DPIv2 [9], or AgentX [10]) to the subagents.

2.3 Interfaces Between Modules

In this section, we describe the communication interfaces between component modules in terms of message format and content. Information exchange between the modules will be done via message passing. Each module would maintain a message queue into which other modules would deposit their messages. Each incoming protocol message is tagged and stored in a message pool for later retrieval. To ensure proper communication among the modules, the message should include its source ID and related authentication information wherever applicable. For instance, if two consecutive modules are executed under one process, then authentication between these two modules may not be an issue. However, if two modules are distributed in two different processes or platforms, then the information of ⟨source_id, signature⟩ should be provided for facilitating authentication.

When a module interfaces with more than one module, it will maintain a separate input queue for each such module. This will enable us to implement priority mechanisms, if so

desired, to process messages from certain modules before others. Also, the input queue will itself serve as a means to identify which module originated the message.

PrevM_Input: The first interface is between network module and JiNao's prevention module. The input message to the prevention module will include the following fields:

```
PrevM_Input{              /* Protocol Data Unit */
   protocol_id;
   PDU itself;
   length of the PDU;
   timestamp of the PDU;
};
```

PrevM2ProEng: If the packet passes the prevention rule checking, it will be sent to protocol engine for execution.

```
PrevM2ProEng{             /* Protocol Data Unit */
   PDU itself;
   length of the PDU;
};
```

PrevM2LDetM: After the routing packet passes through the prevention module, it will be forwarded to detection modules for further examination. The message will include the original routing packet information plus other information rendered by the prevention module. From prevention module to the statistical analysis module, the message format will contain the following information:

```
PrevM2StatM{
   PrevM_Input;
   forwarding_flag;            /* forwarded or not */
   triggered rule object id;   /* triggered rule if applied */
};
```

Similarly, the message format from prevention module to the protocol analysis module will include the following fields:

```
PrevM2ProtM{
   PrevM_Input;
   forwarding_flag;            /* forwarded or not */
   triggered rule object id;   /* triggered rule if applied */
};
```

PrevM2LDecM: The checking result from the prevention module will also be forwarded to local decision module to facilitate decision making.

```
PrevM2LDecM{
   PrevM_Input;
   triggered rule object id;   /* triggered rule if applied */
};
```

LDetM2LDecM: The message received from the prevention module will be processed in parallel by statistical analysis module and protocol analysis module. The analysis results from these two modules will be sent to the decision module to determine if any action needs to be taken. One message format will be defined for each module, respectively. For the interface between the statistical analysis module and local decision module, a message will include the following fields:

```
StatM2LDecM{
   protocol_id;
   timestamp when arriving the decision in the StatM;
   detection output;          /* normal, alert, or alarm */
  <source, detection output type> specific information;
                              /* list of parameters involved in this alarm */
};
```

Similarly, a message from the protocol analysis module to the local decision module has the following information elements:

```
ProtM2LDecM{
   protocol_id;
   timestamp when arriving the decision in the ProtM;
   detection output;          /* Normal, fault or intrusion detected */
  <source, detection output type> specific information;
};
```

The pair of (source, detection output type) provide detailed information to support the detection output rendered by each module. In the case of statistical analysis, this information may include a list of parameters (and their associated ranges) involved in the alarm just issued. For the protocol analysis, this information may be a list of PrevM_Input_PDUs which reflects the sequence of events that are considered attempt of intrusion.

LDecM2PrevM: The message sent from the decision module to the prevention module can be used to insert and delete, or activate and deactivate a certain rule.

```
LDecM2PrevM{
   command;                   /* insert, delete, activate, deactivate, etc. */
   command dependent information;
   rules involved ;
   timestamp of this message;
};
```

The field under command dependent information may be related to setting a new threshold or choosing a specific mode of a parameter.

LDecM2LDetM: Similar to the case above, we will be able to dynamically modify the range of certain parameter in the statistical module or import some new detecting sequences into the protocol analysis module. For a message from the decision module to the statistical module, it has the following fields:

```
LDecM2StatM{
   command;
   command dependent information;
   timestamp of this message;
};
```

where the command dependent information may include an array of parameters and their associated ranges. Similarly, from the decision module to the protocol analysis module, a message will include:

```
LDecM2ProtM{
   command;
   command dependent information;
   timestamp of this message;
};
```

The command dependent information may include a table of finite state machine which representing a new detecting sequence.

LDecM2IAM: From the decision module to the information abstraction module, the message will contain the following elements:

```
LDecM2IAM{
   LDecMInput;          /* PrevM2LDecM, StatM2LDecM, or Prot2LDecM */
   Type of decision     /* normal, fault, or intrusion */
   decision timestamp;
}
```

LDecM2ProEng: This interface would most likely take the form of a function call. The function call would provide a mechanism to query and set various protocol defined options. The exact format would, in general, depend on the specific protocol implementation.

LDecM2SO: This communication from the decision module to the security officer would take the form of either notification via electronic mail or through a graphical user interface. In the latter, the GUI would permit the display of an appropriate alarm/alert message along with other relevant information.

IAM2LDecM: From the information abstraction module to the decision module, the message will have the following format:

```
IAM2LDecM{
   message timestamp
   destination module   /* stats, protocol, prevention, protocol engine */
   type of global info  /* intrusion info, configuration info */
   type specific data   /* intrusion/fault, scope of impact, actions/command,
                        new rule set, add/remove rules */
}
```

IAM2MIB: From the information abstraction module to JiNao MIB, the message includes the following information:

```
IAM2MIB{
   aggregate flag
   /* when no aggregation, flag=0 */
   {
      local decision timestamp
      decision input info    /* the corresponding LDecM input */
      decision output info   /* normal, fault, or intrusion */
   }
   /* with aggregation, flag=1 */
   {
      time interval          /* during which the observation holds */
      local decision timestamp
      decision input info    /* the corresponding LDecM input */
      decision output info   /* normal, fault, or intrusion */
   }
   decision scope of impact
}
```

More discussion about the scope of impact will be given in the section of functional description (Section 4.1.5.3).

MIB2IAM: From JiNao MIB to IAM, the MIB information includes:

```
MIB2IAM{
   global detection results{
      type of detection      /* intrusion/fault etc*/
      scope of impact
      decision source ID
      source Signature
   }
   management application commands   /* retrieve log, interface up/down */
   configInfo{
      source ID
      source signature
      destination module
      config commands         /* activate/deactivate, new rule set,
                                 add/remove rules */
   }
}
```

3 Design Objectives

In this section we discuss the objectives used to guide the system design decision. In general, we want to limit the amount of audit records collected and processed such that the overhead of intrusion detection can be kept to minimum. The detection capabilities will be provided in a non-intrusive way to the target protocols' operation. To avoid degradation of routing and management services, it is desirable to be able to deactivate an intrusion detection process if necessary. Also, we design the agent software in a modular fashion, so that any modification and functional extension can be handled with a minimum effort.

In order to correctly interpret and utilize this design document, it is helpful to clearly understand the basic assumptions on the target networking environment and the exact scope of the project.

Target Environment: The intrusion detection solution prototyped in this project can be applied to any network environment that uses OSPF routing protocol. Examples of such environments are networks consisting of only OSPF-based IP routers, networks containing autonomous systems that are using OSPF protocol, and networks including ATM and IP-Switching technologies but uses OSPF at the IP level. The threat model assumes that some OSPF routing entities may get compromised in ways that they will mis-behave and consequently may disrupt the routing service in the network. The networking hardware and software components may also contain fault conditions which can manifest during the network operation. Since the way these fault conditions manifest are generally unknown beforehand, the best we can aim for in intrusion detection is to be able to detect such manifestations, particularly when they pose a threat to the routing services. Link encryption may or may not be implemented for these routing entities. As long as we observe the behavior of routing information exchange at the routing protocol level, the link level encryption should be irrelevant. Finally, our solution does not assume global collaboration from end hosts. Although we recognize that some orchestrated intrusion attacks cannot be detected without observation from end systems, it is unrealistic to count on global corporations from end systems.

Scope of the Current Project: Our solution addresses the issue of how to quickly and accurately detect conditions in the routing infrastructure that either are already causing disruptions to the routing services or are considered to have the potential to cause disruptions. Without implementing the global detection modules (which are part of the Optional Tasks of this project), detectable conditions will be confined to those that manifest on a local scale, specifically, those that can be observed somehow by neighboring entities. This project does not address the issue of host intrusions, e.g. break-in to a routing entity. While better host intrusion detection and security protection will reduce the chance of a routing entity being compromised, there are always other means of attack, e.g. social attack via a compromised system administrator, that can lead to compromises of these entities. However, such compromises are within our threat model and our solution will be able to help detect them. Finally, the deployment of the intrusion detection system does not require installation of Ji-Nao modules to each every routing entity in the network in order to operate, although wider deployment generally affords better overall detection capability.

3.1 Comprehensiveness

In terms of protecting system from intrusion attacks, it is very desirable to be able to detect different kinds of attacks, both known system vulnerabilities and exploitation on unknown vulnerabilities. We also would like to design a detection system which covers intrusive attempts in various time scales. The employment of the rule-based, protocol-based, and statistical-based approaches in our system fits in these criteria very well due to the complementary nature of their detection mechanisms. While the rule-based and the protocol-based approaches are meant to detect attacks on known security weaknesses, the statistical-based approach is able to catch attempts of exploitation on unknown vulnerabilities. In the meantime, while statistical analysis usually requires a learning (or adapting) period, protocol-based and ,especially, rule-based approaches have a very short response time.

3.2 Scalability

Even though the current scope of the project focuses on the development of local detection capabilities, we expect the system design can be easily extended to a regional and even a more global level. While it is not within the scope of this project, we expect the detection/analysis functions implemented in the local subsystem can be extended to a global level and correlate intrusion events among several routers. The extension to a global level can be hierarchical where several regional management stations can aggregate their detection information to a higher level for establishing a global view of the routing domain status. The communication of this extension can be provided through SNMP ManagerToManager operations.

3.3 Interoperability

We expect that an intrusion detection system will be part of the network management framework in order to best capitalize its benefits. SNMP is a network management information exchange protocol that has been implemented and widely deployed in the existing networks. Since SNMP is the industrial *de facto* standard, our system will be able to integrated with other SNMP-based system or security applications with relative ease.

Another aspect of the interoperability (including module reusability) is related to the questions of

1. identifying well-understood building blocks/modules for the intrusion detection systems,
2. clearly defining the functionality and interfaces for these modules, and
3. defining the basic protocol for inter-operation among the appropriate modules.

Currently, these is a joint effort to address the system interoperability and module reusability issues among DARPA/ITO sponsored projects, especially among the intrusion detection community, in order to fully take advantage of the investment and bring forth a better synergistic effect. Our architectural design is consistent with the objectives of this joint effort. We strive to clearly define the common modules (interception, detection/analysis, decision, and management and agent) by specifying their functions and interfaces. As a

member of the CIDF (Common Intrusion Detection Framework) working group, we expect that, as the CIDF joint effort progress further, necessary modifications can be made to further enhance the interoperability and reusability of our system with other systems.

4 Functional Description

In this section, we will describe in detail the functional modules mentioned in Section 2.2.

4.1 Local Subsystem

A JiNao local subsystem logically resides in a router or just next to it. It consists of the following modules: interception/redirection module, rule-based prevention module, protocol and statistical-based detection modules, decision module, and information abstraction module. It also includes a management information base (MIB) and remote MIB agent functions which provide access to remote management applications.

4.1.1 Interception/Redirection Module

The responsibility of the interception module is to *redirect* the target protocol information flow to the prevention layer. Also, if possible, the redirected information flow should be timestamped. Depending on the target protocols under JiNao's protection, interception module can be placed in multiple protocol layers (Figure 2):

IP/IPSEC: The PDUs can be intercepted at the IP layer. In many operating systems (e.g., Linux and BSD), the kernel-level IP packet interception has been supported. For example, the *ipfwadm* package is for flexible implementation of firewall mechanisms in the kernel. If certain IPSEC options are turned on, we should redirect the PDUs after the security checks performed by the IPSEC layer. This will eliminate immediately PDUs violating the protection provided by IPSEC.

Device Driver: Network hardware device driver (*e.g.,* EtherNet device driver) is usually not a good place for interception because IP packets can be fragmented (large PDUs), encapsulated (tunneled PDUs), and authenticated/encrypted (IPSECed PDUs). Performing interception at this level may introduce unnecessary system complexity and should be avoided unless the redirection function is unavailable in all other layers.

Higher-Layer Protocols: Sometimes, it is necessary to perform interception in layers beyond IP. For example, in protecting SNMP, (especially SNMPv2 and v3), the SNMP PDU might be encrypted. Under this case, we should intercept the PDU flow after the authentication and decryption process. One important requirement for this approach is the availability of an interception interface (something like ip_firewall mechanism but in a higher layer). If this is not possible, likely we need the source code of the target protocols so we can build one ourselves.

Notice that, for many existing protocols (like SNMPv1 or OSPFv2), normally PDUs are not encrypted. Therefore, interception/redirection in the IP/IPSEC layer will provide

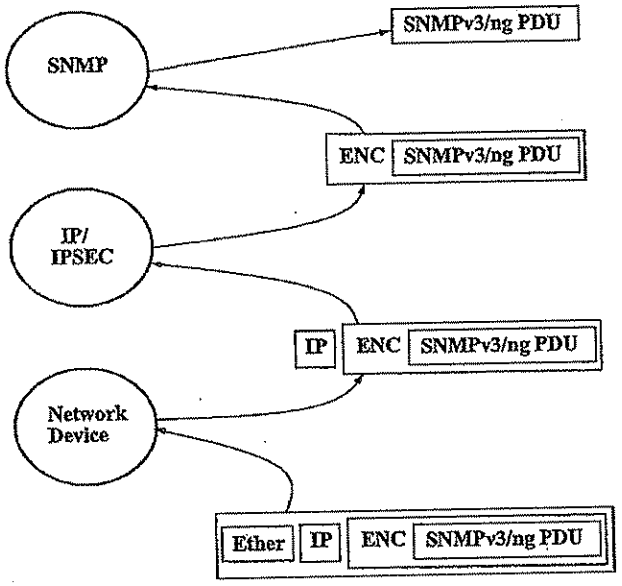


Figure 2: Packet formats across protocol layers.

enough protocol information for the IDS. However, for newer (in progress) protocols (like SNMPng/v3), encryption is an implementation-mandatory option (*i.e.*, it is an option but its implementation for a SNMPng/v3 engine is mandatory). Very little valuable information can be derived from the IP/IPSEC layer.

Here we would like to bring up an important suggestion: the standard committee for those protocols (assuming still in progress) **should** consider the interception interface in the protocol. For example, the SNMPng/v3 working group under IETF just started, and the current framework document drafts did not consider intrusion detection or application-layer firewall mechanisms at all. After the deployment of SNMPng/v3 products, it will be very difficult to consistently intercept SNMP PDUs for either "active/real-time" intrusion detection or firewall protection. Currently, the SNMPng/v3 working group is considering the logging facility, which is useful for passive/offline intrusion detection. We feel that simple logging is not sufficient to protect the target protocols. This principle should apply to not only SNMPng/v3 but also all in-progress networking protocols that we are potentially interested in protecting.

In case of routing services, since OSPF protocol runs directly over IP, we will either place our interception function within IP in the kernel space or implement it under OSPF in the user space. As we mentioned earlier, the interception and prevention modules together are acting like a firewall to filter out any packet with clear security violation. Putting this firewall right within IP (in the kernel space) allows us to protect a set of applications (e.g., GateD, SNMP, HTTP) without modifying their source codes. It can be done through the notion of dynamic module loading supported by Linux and BSD. It is quite powerful and flexible. We intend to experiment both approaches.

4.1.2 Prevention Module

Prevention module is the bridging component between the JiNao/IDS agent and the target protocols protected under JiNao. On one hand, prevention module is virtually inserted into the raw/original target protocol information flow to intercept and screen protocol data units (PDU). On the other hand, prevention module offers JiNao-formatted information for the detection module (protocol and statistical analysis). The prevention module also is programmable and has a control interface for activation/deactivation and dynamic configuration.

The prevention module is separated further into two different sublayers: *prevention layer* and *extraction layer* as shown in Figure 3.

4.1.2.1 Prevention Layer The redirected PDU flow from the interception module is the input for the prevention layer. The prevention layer's main objective is to decide whether a PDU should flow back to the target protocol engine or not. A small number of rules will be used to check against the PDU flow. For example, a rule could be specified such that all OSPFv2 PDUs with originator address XYZ received from the eth2 interface should not be forwarded to the OSPF engine (*e.g.* GateD program).

Quick decision about forwarding a PDU or not is a key design objective for this layer. That is the main reason we purposely put this layer under the extraction. This rule set will examine the raw information from the interception module and immediately forwards the valid PDUs back to the target protocol engine. Otherwise, the target protocol engine might observe a significant delay in receiving the PDUs. On the other hand, in real implementation, these two layers and the interception module can be merged into one for better performance.

The decision module will interact with this layer for controlling the rule set. These rules can be dynamically loaded/unloaded and/or activated/deactivated. This flexible control interface between the decision module and the prevention layer will enable *run-time objective-driven* prevention.

Notice that sometimes a rejected PDU (*i.e.*, not being forwarded) might still be interesting to the detection module. Therefore, the rule must specify two things:

1. Should the PDU be forwarded to the protocol engine?
2. Should the PDU be forwarded to the detection module?

If the answer to the first question is YES, the original PDU will be sent back to the protocol engine. If the answer to the second question is YES, a copy of the PDU will be passed to the extraction layer.

4.1.2.2 Extraction Layer The information expected by the detection module should follow the same JiNaoPDU formats as we describe before. Therefore, in this layer, the main objective is to transform the raw information into the JiNao Information PDUs that can be accepted by the detection module through a generic interface.

Sometimes, it might be necessary, in this layer, to correlate multiple different raw PDUs (from multiple interception points) and generate only one single JiNao PDU. One practical example will be related to information regarding the hardware interface that an PDU is

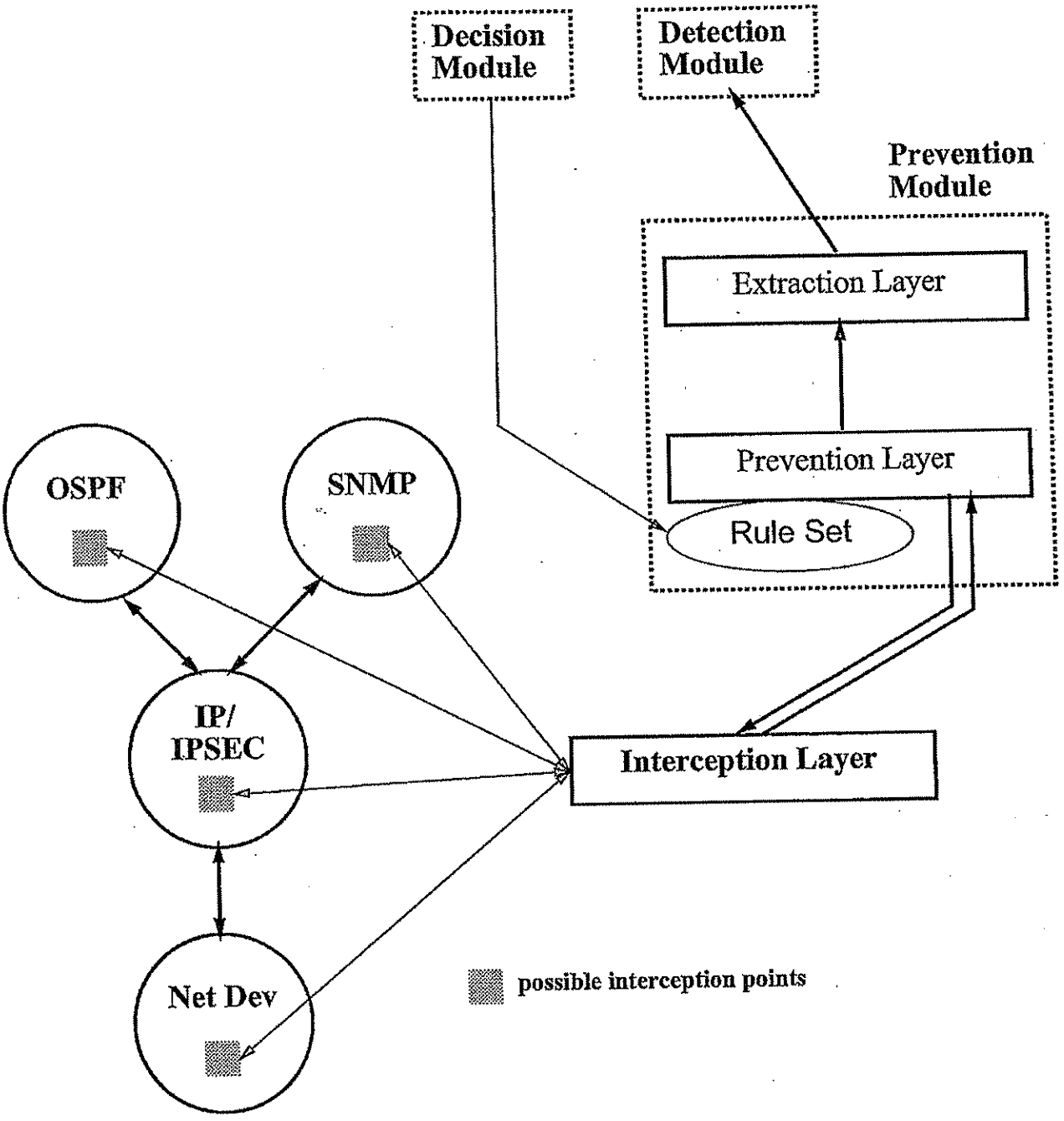


Figure 3: Interception and prevention modules and their relationship.

received. We feel that this information is valuable to detect certain spoofing attacks, but how to collect is a little tricky. Depending on the network protocol stack implementation, sometimes the information is there in the IP layer. However, on the same node, an IP packet might travel through more than one device interface. For example, an encapsulated packet will pass through both /dev/eth and /dev/tunnel, and in this case, we need to record both interfaces for this particular packet. Furthermore, as we mentioned before, application-layer encrypted PDUs should be intercepted at the application layer. In the application layer, most liekly, the device interface information for the decrypted PDUs has already lost. To recover the device interface information (*e.g.,* this SNMPng/v3 PDU is from eth1 and tunnel3), we need to, possibly, intercept the encrypted version of the same packet in another interception point (*e.g.,* IP). This two interceptions represent logically a single PDU. Therefore, they should be correlated and treated as one JiNaoPDU.

4.1.3 Detection Module

After the incoming packet passes through the rule-based checking, it will be forwarded in parallel both to the protocol engine for execution and to the detection module for further analysis.

4.1.3.1 Statistical Analysis Module In the area of computer security, statistical analysis has been reported in various projects in the literature, for example, the NIDES project at SRI [11], Wisdom and Sense at Los Alamos National Laboratory [12], and Haystack project [13] at Haystack Laboratories. Among these examples, the NIDES project at SRI is most extensive in its scope and development. It also has the most complete documentations available to the general public. With the understanding of statistical analysis's general applicability, we will adapt NIDES's statistical algorithm in our approach as a starting point and modify it as necessary.

The basic statistical approach is to compare a subject's short-term behavior with the subject's historical or long-term behavior. A subject is context-dependent, which can be a user of a computer system, a credit card holder, or one of the neighbor routers in the case of this project. In comparing short-term behavior with long-term behavior, the statistical component is concerned both with long-term behaviors that do not appear in short-term behavior, and with short-term behaviors that are not typical of long-term behavior. Whenever short-term behavior is sufficiently unlike long-term behavior, a warning flag is raised. In general, short-term behavior is somewhat different from long-term behavior, because short-term behavior is more concentrated on specific activities and long-term behavior is distributed across many activities. To accommodate this expected deviation between short-term and long-term behavior, the statistical component should account for the amount of deviation that it has seen in the past between a subject's short-term behaviors and long-term behaviors. The statistical component issues a warning only if the current short-term behavior is very unlike long-term behavior relative to the amount of deviation between these types of behaviors that it has seen in the past. This feature will be revisited later when we explain the computational aspect of the algoritm. In the following sections, we introduce the major components of the algorithm, and describe its scoring statistics and computational process.

4.1.3.1.1 Components of the Statistical Approach In this section, we introduce some major components of the statistical approach which includes various measures, half-life in updating both short-term and long-term probability distributions, scoring statistics, and computational algorithm in obtaining these statistics.

- Measures: Aspects of subject behavior are represented as measures (e.g., packet and LSA arrival frequencies in terms of their types or sources). For each measure, we will construct a probability distribution of short-term and long-term behaviors. For example, for the packet types received, the long-term probability distribution would consist of the historical probabilities with which different types of packets have been received, and the short-term probability distribution would consist of the recent probabilities with which different types packets have been received. In this case, the categories to which probabilities are attached are the names of packet types, which are learned by the system as they are received. We would classified the Ji-Nao measures into two groups: activity intensity and audit record distribution measures. These two types of measures serve different dimensional purposes. The activity intensity measures determine whether the volume of general activity generated in the recent past (depending on the half-life of the measure, here "recent past" corresponds to the time span of last several half-lifes) is normal. These measures can detect bursts of activity or prolonged activity that is abnormal, primarily based on the volume of audit data generated. The audit record distribution measure determines whether, for recently observed activity (say, the last few hundred audit records received), the types of actions being generated across neighbors are normal. For example, we might find that the last 200 routing packets received contained 120 of Hello packets, 15 of Database Description packets, 10 of Link State Request packets, 35 of Link State Update packets, and 20 of Acknowledegment packets. These data are compared to a profile of previous activity (generated over the last few months) to determine whether or not the distribution of activity types generated in the recent past (i.e., the last few hundred audit records) is unusual.

- Half-life: The specification of a half-life determine the number of audit records or days of audit record activity that constitute short-term and long-term behavior. For the long-term probability distributions, the experience from NIDES suggests to set the half-life at 30 profile updates, which are typically performed once daily. With this setting, audit records that were gathered 30 updates in the past contribute half as much weight toward the probability distribution as do the most recent records. Audit records that were gathered 60 updates in the past contribute one-quarter as much weight, and so forth. Thus, the most recent days of activity contribute more than the more distant days of activity, and eventually the long-term profile "forgets" about very distant behavior. For the long-term profile, the long-term aging factor is applied to the historical data at each update, and then the new information is folded in. For the short term profile, the short-term aging factor is applied to the profile with each audit record and the current audit record is folded in.

- The S and T^2 statistics: For each audit record generated by a subject, the statistical module generates a single test statistic value, denoted T^2, that summarizes the degree of abnormality in the subject's behavior in the near past. The T^2 statistic is itself a

summary judgement of the abnormality of many measures taken in aggregate. Suppose that there are n such constituent measures denoted by S_i, $1 \leq i \leq n$. Each S_i is a measure of the degree of abnormality of behavior with regard to a specific feature. The T^2 statistic is set equal to the sum of the squares of the S_i:

$$T^2 = (S_1^2 + S_2^2 + ... + S_n^2)/n$$

- The Q Statistic: The degree of difference between a long-term profile and short-term profile for a measure is quantified using a chi-square-like statistic, comparing observation (the short-term profile) to expectation (the long-term profile). The resultant numerical value is call Q in NIDES. Each S measure is derived from a corresponding Q statistic. In fact, each S measure is a "normalizing" transformation of the Q statistic so that the degree of abnormality for different types of measures can be added on a comparable basis. Two different methods for transforming the Q statistics into S values are used. One method is used for computing the values fo S corresponding to intensity measures; a second method is used for computing the values of S corresponding to audit record distribution measures. The computation of Q statistics and the transformations from Q to S statistics will be explained in the following sections.

4.1.3.1.2 Computing the Q statistics for the intensity measures When a neighbor router is first brought up and audited, that router has no history. Consequently, we must choose some convenient value to begin the Q statistic history. For instance, we might initially let each Q measure be zero, or some value close to the mean value for other routers.

Each Q statistic for intensities is updated each time a new audit record is generated. Let Q_n be the value for Q after the n^{th} audit record, and Q_{n+1} be the value for Q after the $(n+1)^{th}$ audit record. The formula for updating Q is:

$$Q_{n+1} = 1 + 2^{-r(t_{n+1}-t_n)}Q_n$$

where

- The variable t_n represents the timestamp of the n^{th} audit record.
- The decay rate r determines the half-life of the measure Q. Large values of r imply that the value of Q will be primarily influenced by the most recent audit records. Small values of the decay rate r imply that Q will be more heavily influenced by audit records in the more distant past. For example, a half-life of 10 minutes corresponds to an r value of 0.1 ($r = -[log_2(0.5)]/10$). The security officer may set the half-life of the intensity measures at any values that he or she feels appropriate.

Q is the sum of audit record activity over the entire past activities, exponentially weighted so that the more current activity has a greater impact on the sum. Q is more a statistic of near past behavior than of distant past behavior. One important property of this Q statistic is that it's not necessary to keep extensive information about the past to update Q.

Noted that the intensity measures use clock time as the unit by which age is calculated. This is important because the intent of this measure is to assess the extent to which bursts of activity are normal.

4.1.3.1.3 Computing the Q statistics for the audit record distribution measures Suppose that we have established M activity types. For each activity we must calculate a long-term historical relative frequency of occurrence, denoted f_m, for that activity type. For instance, suppose that over the last six months, 35% of all received packets are Hello packet type. Then f_m for the Hello packet type would be 0.35.

The algorithm used to compute f_m on the k^{th} day is equal to

$$f_{m,k} = (1/N_k) \sum_{j=1}^{k} W_{m,j} 2^{-b(k-j)}$$

where b is the decay factor similar to r which was defined before, and $W_{m,j}$ is the number of packets received on the j^{th} day that indicate that the m^{th} packet type was received. N_k is the exponentially weighted total number of audit records that have occurred since the router was first monitored. The formular for N_k is:

$$N_k = \sum_{j=1}^{k} W_j 2^{-r(k-j)}$$

where W_j is the number of packets received on the j^{th} day.

The Q statistic compares the short-term distribution of the types of packets that have been received with the long-term distribution of the same types. In the simplest situation, Q_n (the value of the Q statistic when the n^{th} packet is received) is defined as follows:

$$Q_n = \sum_{m=1}^{M} [(g_{m,n} - f_m)^2 / V_m]$$

where $g_{m,n}$ is the relative frequency with which the m^{th} packet type has been received in the recent past (which ends at the n^{th} packet), and V_m is the approximate variance of the $g_{m,n}$.

If we view $g_{m,n}$ as the short-term profile for the audit record distribution and f_m as the long-term profile for audit record distribution, then Q_n is larger whenever the distribution of packet types in the recent past differs substantially from the historical distribution of packet types, where "substantially" is measured in terms of the statistical variability introduced because the near past contains relatively small sample size. The value of $g_{m,n}$ is given by the formula

$$g_{m,n} = (1/N_r) \sum_{j=1}^{n} [I(j,m) 2^{-r(n-j)}]$$

or by the recursion formular

$$g_{m,n} = 2^{-r} g_{m,n-1} + [I(n,m)/N_r]$$

where $I(j,m) = 1$ if the j^{th} audit record indicates packet type m has been received and 0 otherwise. The decay rate r for Q determines the half-life for the Q statistic. N_r is the sample size for the Q statistic, which is given by the formula

$$N_r = \sum_{j=1}^{n} 2^{-r(n-j)}$$

The value of V_m is given by the formula $V_m = f_m(1 - f_m)/N_r$.

4.1.3.1.4 Computing the frequency distribution for Q The first step in calculating the historical probability distribution for Q is to define bins into which Q can be classified. We will use 32 bins for a Q statistic. Let Q_{max} be the maximum value that we ever expect to see for Q. This maximum value depends on the particular types of measures being considered. The cut points for the 32 bins are defined on either a linear or geometric scal. For example, when a geometric scale is used, bin 0 extends from 0 to $Q_{max}^{1/32}$, bin 1 extends from $Q_{max}^{1/32}$ to $Q_{max}^{2/32}$, and bin 31 extends from $Q_{max}^{31/32}$ to infinity.

Let P_m denote the relative frequency with which Q is in the m^{th} interval (bin). Each Q statistic is evaluated after each packet is received (whether or not the value of Q has changed). The formula for calculating P_m on the k^{th} day after a router was brought up is:

$$P_{m,k} = (1/N_k) \sum_{j=1}^{k} W_{m,j} 2^{-b(k-j)}$$

where k is the number of days that have occurred since the router was first monitored; $W_{m,j}$ is the number of audit records on the j^{th} day for which Q was in the m^{th} bin; N_k is the exponentially weighted total number of audit records that have occurred since the router was first monitored.

The formula for N_r is:

$$N_k = \sum_{j=1}^{k} W_j 2^{-b(k-j)}$$

where W_j is the number of packets received on the j^{th} day.

The computations for $P_{m,k}$ and N_k can be simplified by using the following recursion formulas:

$$\begin{aligned} P_{m,k} &= (2^{-b} P_{m,k-1} N_{k-1} + W_{m,k})/N_k \\ N_k &= 2^{-b} N_{k-1} + W_k \end{aligned}$$

$P_{m,k}$ and N_k is updated once per day and we keep running totals for $W_{m,k}$ and W_k during the day.

4.1.3.1.5 Deriving S from Q for the intensity measures For the intensity measures, the value of Q corresponding to the current packet represents the number of packets received in the recent past. Here, "recent past" corresponds to the last few minutes for the Q statistic with a half-life of one minute and to the last several hours for the Q statistic with a half-life of one hour. In addition to knowing the current value for Q, the statistical module maintains a historical profile of all previous values for Q. Thus, the current value of Q can be compared to this historical profile to determine whether the current value is anomalous.

The transformation of Q to S for the intensity measures requires knowledge of the historical distribution of Q which can be obtained from Section 4.1.3.1.4. For example, we might find the following historical distribution for the intensity measures Q with a half-life of one minute:

- 1% of the Q values are in the interval 0 to 10 packets

- 7% are in the interval 10 to 20
- 35% are in the interval 20 to 40
- 18% are in the interval 40 to 80
- 28% are in the interval 80 to 160
- 11% are in the interval 160 to 320

The S statistic would be a large value whenever the Q statistic was in the interval 0 to 10 or was larger than 320 (either because it is a relatively unusual value for Q or the value has not occurred historically). The S statistic would be close to zero whenever Q was in the interval 20 to 40, because these are relatively frequently seen values for Q.

The algorithm for deriving S values from Q statistics for the intensity measures is as follows:

1. Let P_m denote the relative frequency with which Q belongs to the m^{th} interval. Using the previous example, the first interval is 0 to 10 and the corresponding P value (P_0) equals 1%. There are 32 values for P_m, with $0 \leq m \leq 31$.

2. For the m^{th} interval, let $Tprob_m$ denote the sum of P_m and all other P values that are smaller than or equal to P_m in magnitude. In our example, $Tprob$ for the interval of $40 \leq Q \leq 80$ equal to 37% (18% + 11% + 7% + 1%).

3. For the m^{th} interval, let s_m be the value such that the probability that a normally distributed variable with mean 0 and variance 1 is larger than s_m in absolute value equals $Tprob_m$. The value of s_m satisfies the equation

$$P(|N(0,1) \geq s_m) = Tprob_m$$

or

$$s_m = \Phi^{-1}(1 - (Tprob_m/2))$$

where Φ is the cumulative distribution function of a N(0,1) variable. For example, if $Tprob_m$ is 5% then we set s_m equal to 1.96, and if $Tprob_m$ is equal to 10%, then we set s_m equal to 1.28.

4. Suppose that after processing an audit record we find that the Q value is in the m^{th} interval, then S is set equal to s_m.

4.1.3.1.6 Deriving S from Q for the audit record distribution measure For audit record distribution measures, Q compares short-term behavior to long-term behavior and measures the extent to which the composition of the most recent few hundred records is consistent with long-term composition.

Like intensity measures, we calculate a long-term profile for Q using 32 intervals for the audit record distribution. The range of the Q values is expressed in terms of the degree of similarity between the short-term profile and long-term profile with larger numbers representing less similarity.

Because of the difference in the way that Q is defined for intensity measures and audit record distribution measures, the transformation of Q to S is slightly different for audit record distribution measures. Let $Tprob_m = P_m + P_{m+1} + \dot{+}P_{31}$. In our previous example, the $Tprob$ value of the interval $40 \leq Q \leq 80$ would be equal to 18% + 28% + 11% = 49%. Thus, in these cases, S is a simple mapping of the percentiles of the distribution of Q onto the percentiles of a half-normal distribution.

In practice, the Q tail probability calculation is done only once at the update time (daily). Each interval for Q is associated with a single s value, and when Q is in that interval, S takes the corresponding s value.

4.1.3.1.7 Training and updating Training is the process by which the statistical component learns normal activity for a subject. It consists of C (category) training (wherein the component learns the observed categories for each measure), Q training (wherein the system builds an empirical distribution for the Q statistic, which measures the measure-by-measure difference between the long- and short-term profiles), and T training (wherein the system establishes the threshold for the measure statistic, which is collected across all active measures). All three phases have a minimum training period before anomaly scoring begins. Training continues in the steady state, permitting a degree of adaptation to new behavior of a subject.

Initially the component is in training because long-term profiles are being created. A new profile (long-term and short-term) is created whenever a new subject is first encountered. The statistical analysis module will continue to train by recording and updating a subject's behavior in the subject's long-term profile. A subject's long-term profile is considered trained when at least one measure has gone through the C, Q, and T training phases. At this point anomalies may be reported. According to NIDES's experience, the number of updates required to complete each training phase is the training period (by default 20 updates) divided by the number of phases (3) and rounded up the nearest whole number. By default each training phase, C, Q, and T requires 7 updates to complete.

4.1.3.2 Protocol Analysis Module

4.1.3.2.1 Overview The Protocol Analysis Module (PAM) uses message traffic and knowledge about the protocol engine to detect when an intruder is attempting an attack. When the PAM detects such an attack, it sends an alarm message to the Local Decision Module describing the attack and containing the sequence of messages used in determining that the attack took place.

4.1.3.2.2 Interface The PAM serves as a "stream processor": it accepts a stream of inputs and delivers a stream of outputs.

Input As a submodule of the Local Detection Module, the PAM accepts two kinds of input packets: `PrevM2LDetM_PDU` and `LDecM2ProtM`. These two kinds of inputs require different responses.

- Packets of form `PrevM2LDetM_PDU` come from the Local Prevention Module (LPM) and contain a message from the network together with a flag indicating whether or not the LPM forwarded the message to the protocol engine or dropped it. The PAM uses these packets to track the possibility of intrusions.
- Packets of form `LDecM2ProtM` come from the Local Decision Module (LDecM) and contain commands to alter the intrusions being tracked by the PAM. In the current design, these commands will consist of requests either to add or to remove finite-state machines from the collection maintained by the PAM; this issue is explained in more detail below.

Output The PAM generates output packets of type `ProtM2LDecM`. The fields in these packets contain the following.

- `protocol_id`: an indicator of the protocol for which an intrusion has been detected (in general, Ji-Nao will be able to detect intrusions in several different protocols simultaneously).
- `alarm`: an indicator of the type of intrusion detected.
- `pdu_list`: a list of (pointers to) messages that led to the detection of the intrusion.

4.1.3.2.3 Implementation

Overview The PAM will maintain a collection of finite-state machines (FSMs) for each protocol JiNao is capable of monitoring. Each FSM will be used to detect one kind of intrusion and will be constructed off-line in a manner described later in this section.

The control flow of the system is as follows. Upon receiving an input `I`, the PAM will first examine its type. If `I` is a `PrevM2LDetM_PDU` packet then the PAM will do the following.

1. If the forwarding flag in `I` is "false", no further processing will be performed because this packet was not forwarded to the protocol engine.
2. If the flag is "true", then the PAM will perform the following for each FSM it is currently maintaining.
   (a) The FSM's current state will be updated based on the form of the network message contained with `I`. If the state changes, `I` will be inserted into the FSM's message queue.
   (b) If the FSM's new state is its initial state, then the queue will be flushed (there is no attack underway when this is the case).
   (c) If the FSM's current state indicates that an attack has taken place, then:
      i. An output packet `O` of type `ProtM2LDecM` is generated and fields initialized as follows.
         - The protocol identifier in `I` is copied into `O`.

- The detection output is initialized with the kind of intrusion detected.
- The message list in `O` is set to the contents of the FSM's queue.

ii. The message queue is flushed.

iii. The FSM's state is reset to its start state.

iv. `O` is output.

On the other hand, if `I` is of type `LDecM2ProtM` then the PAM will engage in the following.

1. If the command contained in `I` is a `delete` command, then the command-specific information includes a descriptor for one of the FSMs maintained by the PAM. In this case the relevant FSM (and its message queue) are removed.

2. If the command is an `insert` command, then the command-specific information includes a descriptor for a new FSM to be maintained by the PAM, together with a protocol identifier indicating which protocol this FSM is intended to be associated with. In this case the FSM is inserted into the list of FSMs the PAM maintains, together with a new message queue for this machine.

If the commands contain invalid arguments then they are ignored.

Example To illustrate the "intrusion-tracking" the PAM will undertake, consider the following attack that can arise during the adjacency establishment phase in the OSPF protocol. In this phase of the protocol two routers attempt to establish an adjacency relationship that will eventually be broadcast to the relevant parts of the rest of the network. After ensuring that each other is up and capable of communicating (using a Hello protocol), the routers negotiate their respective master/slave status and a sequence number. The master then transmits its routing database to the slave using of Database Description (DD) packets, each of which is tagged with a sequence number and each of which the slave must acknowledge.

One attack would involve an intruder attempting to masquerade as a master with the intent of corrupting the (existing master's) database. If this attempt occurs after the negotiation of the master/slave relationship, then this attempt can be detected. The relevant FSM for detecting this intrusion would have the following states and behave as follows.

Down This is the start state and represents a situation in which no master/slave relationship exists. All inputs are ignored except those involved with the Hello protocol.

Attempt, Init, 2-Way States associated with the Hello protocol; the behavior of these states is exactly as described in OSPF definition.

Exstart In this state, the arrival of a DD packet from the neighbor is analyzed to determine whether the neighbor or this router should be the master. If this router, enter state **Master**; otherwise, enter state **Slave**.

Master In this state, all incoming DD packets should be acknowledgements. This can only fail if the sequence number in the incoming packet fails to match the current sequence number. In this case, enter **Alarm**.

Alarm An intrusion has been detected.

A couple of things should be noted about this example. Firstly, we require one FSM for each potential adjacent router (so each FSM detects an intrusion of the form "router R is corrupted"). Secondly, the description makes implicit use of counters that are difficult to encode in pure FSMs. This will have implications for our representation of FSMs given below.

Details One of the chief goals of the Ji-Nao project is that Ji-Nao should provide extensibility: it should be easy to adjust the kinds of intrusions that are tracked. This requirement has the following implications for the PAM.

1. The PAM should be reconfigurable at run-time: in particular, users should be able to add (and remove) FSMs as new types of intrusions become of concern (and old types cease to be).
2. Adding FSMs should not require recompilation of the PAM.

To accommodate these concerns, we envisage a table-driven implementation of FSMs together with a generic driver routine. Adding a new FSM then amounts to defining a table for it and then loading it into the PAM; the driver routine (which will be compiled into the PAM) would then handle the "execution" of the FSM. We describe each of these concepts in turn.

In a traditional tabular representation of a FSM, each row in the table represents a state, and each column represents an input. If the $(i,j)^{th}$ element of such a table is k, this means that if the FSM is in state i and input j arrives, the new current state should be k. States are usually encoded as integers in the range $0,\ldots,N-1$, where N is the total number of states.

For efficiency reasons, we propose a modification of this scheme. As in the usual case, states will be represented using integers in the range $0,\ldots,N-1$, but rather than associating a row in a table with each such integer, we will instead associate a (pointer to) a function. The function will take as input a network message and compute the new state to transition to. This will allow FSMs to be represented internally as arrays of pointers to functions.

The driver routine will maintain a linked list, each cell of which will contain the following.

- A protocol id.
- A FSM.
- An integer containing the current state of the FSM.
- A message queue for the FSM.
- The alarm type tracked by the FSM.

Upon receiving an input I, the driver routine will scan through the linked list. For each cell whose protocol id matches the one contained in I, the driver will look up the function associated to the current state and apply it to the network message in I to compute the new state. If the new state is the same as the old state, processing stops; if the new state differs from the old state, then the current state is updated, and I is inserted into the message queue. If the new state is an alarm state (always assumed to be state $N-1$), then an appropriate output is generated, and the current state is reset to the start state (always assumed to be 0).

In order to add a new FSM, then, a user (i.e. the Local Decision Module) may do the following.

1. Design the desired FSM.
2. Implement the functions used for processing inputs.
3. Store representations of pointers to these functions in a file.
4. Dynamically link the code containing the input-processing functions to the PAM.
5. Send an of type `LDecM2ProtM` containing the machine description to the PAM.

Designing FSMs We now describe how the FSMs used in this module will be designed. We envisage an approach based on abstracting the FSM describing the whole protocol. The general idea is this.

1. Formalize the section of the protocol affected by the intrusion in question as a FSM. This can be done from the protocol definition; indeed, modern protocol standards usually include FSMs in them.
2. Identify the states and messages that would cause the FSM to deviate from "normal functioning"; this is evident from the intrusion, in general.
3. Hide all transitions involving messages that do not appear on a path from the start state to the states mentioned above.
4. Include transitions from the states mentioned above to the "alarm state".
5. For efficiency, minimize the resulting FSM to make it as compact as possible.

To assist in this task we will use the Concurrency Workbench (CWB) [14], a tool developed at NCSU for analyzing the correctness of networks of communicating FSMs (see `http://www4.ncsu.edu/rance/WWW/cwb-nc.html` for more details). For the purposes of this work the CWB provides three capabilities that we plan to use.

Transition hiding Labels on transitions may be "hidden", i.e. converted into empty labels.

FSM minimization FSMs can be minimized to eliminate redundant states.

FSM determinization Nondeterministic FSMs can be converted into deterministic ones.

We propose to use each of these features to produce the FSMs used in the PAM.

4.1.4 Local Decision Module (LDecM)

The LDecM is required in order to correlate the detection information provided by the protocol and statistical analysis modules along with other information that it may possess via interaction with the global detection module and make a decision as to whether an intrusion has taken place.

Functionally, the local decision module interfaces with both the detection modules and the local MIB agent software. It receives input from the detection modules regarding local intrusion activity as inferred by observing the neighbor's behavior. It also interacts with the local MIB agent to gather intrusion detection information obtained from remote JiNao agents and uses it in conjunction with the data reported from the detection modules to make decisions on intrusion. Finally, it also interacts with the protocol engine to take appropriate steps when an intrusion is detected.

4.1.4.1 Functional Description The key functions of the local decision module are:

1. **Make local decisions on intrusion using data from detection modules and information from remote agents:** This is the primary function of the LDecM. As stated earlier, both the protocol and statistical analysis modules look at network behavior independently. Incoming routing traffic is analyzed in these modules for signs of potential fault/intrusion. In many cases, each of these modules may be able to make a decision on intrusion independently. However, there are cases where information from these modules needs to be correlated with global information to make a more informed and accurate decision as outlined in Section 4.1.4.2 below.

2. **Provide information for the IAM:** While it is more efficient to detect intrusions locally, as far as possible, there are cases where only a global agent can make a determination of whether an intrusion has taken place based on information gathered from several local JiNao decision modules. This is accomplished in our system via the use of the specified JiNao MIB. A remote management subsystem will issue SNMP-GET requests for information maintained by the LDecM, and sometimes it might want to receive SNMP-TRAPs when certain events happen. LDecM needs to provide operations to support these requests for all the JiNao MIB variables it owns.

3. **Propagate changes in MIB information, if so indicated, to the detection and prevention modules:** One of the important features of the JiNao system is that it is adaptive to changing network conditions/configurations. This implies that the set of rules upon which the prevention module operates on or the threshold parameters which the statistical module uses to distinguish normal from abnormal behavior or the set of minimum detecting sequences employed by the protocol analysis module may need to be changed dynamically in response to changing network conditions. For instance, should a new point of network connectivity come up, the normal traffic profiles would need to be modified to account for the traffic from the new connection. Should a new attack be discovered that can be prevented by implementing a new set of rules, the rule base for the prevention module would need to be updated. JiNao agents can be updated from a central location via the use of the JiNao MIB. The LDecM will

propagate any information affecting the behavior of various detection modules to these modules upon the requests from the remote security management applications. In addition, based on the decision it arrives at, the LDecM could initiate these changes itself. For instance, if it detected suspicious activity, it could activate additional rules in the prevention module or adjust thresholds in the statistical module.

4. **Inform the local protocol engine in the event an intrusion is detected;** The LDecM interacts with the protocol engine in order to take appropriate action if an intrusion is suspected/detected. For instance, when suspicious activity is detected it may instruct the protocol engine to turn on certain special modes of operation e.g. detailed logging of messages and protocol events, log information relating to route updates and modifications etc..

   The LDecM can also take appropriate defensive measures. This can include turning an interface off when a router connected via that interface has been detected to be faulty/compromised; issuing commands to undo the effects of recent route update messages, if any, from the compromised router etc..

5. **Notify security officer or other appropriate management entity of faults/ intrusion detection:** Upon detection of a fault/intrusion, the LDecM must take steps to notify the appropriate network security personnel. This can be done via the use of a GUI that will make the fault/intrusion information visible on the security personnel's terminal. For less critical events, notification could be performed via electronic mail.

   Periodic log information will be maintained on a regular basis(daily, weekly. monthly etc) which can be reviewed for any suspicious activity.

4.1.4.2 Examples Example of how both protocol and statistical modules might say there is intrusion, when there is none:

Consider a network containing two routers, A and B. Although, part of the same network, routers A and B belong to different regions as far as power supply is concerned. Consider the situation when there is a power outage in a portion of the network affecting router B. There will be no response to any protocol messages and the protocol analysis module in router A will conclude that the neighboring router is either faulty or has been compromised. The statistical analysis module will also notice that the message rate from that particular router has dropped to zero which would be very different from the normal profile for that router. Hence, it too would conclude that the router is faulty/ compromised. In this case, both the protocol and statistical modules would report to the local decision module indicating a fault/intrusion condition.

If however, the local decision module had been made aware by the remote agent of the power outage for the region containing router C, it could look up the region router C belonged to and infer that it was subject to the power outage. Hence it could decide that the apparent anomalous behavior of router B was due to the power outage and ignore the detection information from the protocol and statistical analysis modules. This can be incorporated in software as a set of known existing network faults that must be checked for first, before making a decision on intrusion. In this case, the LDecM can instruct the statistical and protocol analysis modules to cease monitoring router B until further notification.

4.1.4.3 Exceptions and Errors

Exceptions: no acknowledgement from prevention/detection modules in response to a parameter update/create message.

Errors: unrecognized message format message tag points to a non-existent message or is null; rule id reported by prevention module does not exist in decision module's copy of the rule base.

4.1.4.4 Remarks The fact that the local decision module uses information disseminated by the remote JiNao agents in order to make a decision on intrusion leads to a scalable architecture. Indeed, in the converse situation, if the local agents had to forward _all_ their detection information to the global agent in order for the the global agent to make the decision, the global agent would become a centralized decision maker and the architecture would not scale. In our system global information is utilized locally to make a globally aware local decision regarding intrusion. Moreover, the architecture also provides for monitoring attacks which can only be detected at a higher network level. The system is adaptive to changing network conditions in that parameter values/thresholds of various detection modules can be dynamically changed, new rules added or existing rules deleted.

4.1.5 Information Abstraction Module (IAM)

4.1.5.1 IAM Functions The IAM serves as an interface module between the JiNao local intrusion detection subsystem and the remote JiNao modules as well as other network management applications. In propagating local intrusion detection results to the outside, the IAM aggregates local detection results and converts them into the MIB format. In updating the local detection and prevention modules with new rule sets, the IAM receives and processes requests from the remote subsystems through the JiNao MIB interface.

4.1.5.1.1 Local detection information aggregation and MIB-fication The IAM receives the local detection decisions as well as the detection information (based on local observation) from the LDecM. It performs a simple data reduction by using a run-length coding scheme for long sequences of repeated information. The reduced data is then converted into MIB format and put into the JiNao MIB for management applications as well as remote intrusion detection modules. For example, the input from the LDecM includes the following information:

1. Message input from the Detection Modules to the LDecM,
2. Local detection decision (intrusion, fault, normal).

Under normal conditions, there may be many repeated messages and detection decision reporting the normal situation. The IAM will be able to reduce all these repeated messages into a single message indicating that the normal condition lasted for a certain time period. Similarly, reduction is possible with reporting of persistent fault or intrusion conditions.

4.1.5.1.2 Periodic checking and propagation of global information Another potential important role of the IAM is to monitor the new information in remote JiNao MIBs for a global intrusion detection system. Once such information is available, IAM will retrieve them from the remote global JiNaoMIB, convert them into the format expected by the LDecM, and pass them to the LDecM.

4.1.5.2 Interface Mechanisms and Formats Information exchange on the interface between IAM and LDecM is expected to use message queues in the implementation. In particular, each module will have an input queue for every input interface, onto which other modules supplying inputs will deposit messages. The receiving module will remove messages from these queues and act upon them accordingly. For the interface formats, please see Section 2.3.

4.1.5.3 Scope of Impact Representation In Section 2.3, there is an entry under IAM2MIB which conveys the scope of impact from the decision module. More discussion is in order. Scope of impact information is used by a set of distributed JiNao decision modules in order to better enhance the accuracy of intrusion detection decisions. For example:

- a local decision module (LDecM) may use global information on a power outage to reduce its own false alarm rate with respect to a neighbor router if it knows that the neighbor falls within the scope of impact of the outage;
- a higher-level decision module, i.e. one that has access to observations on a larger topological region, can correlate multiple detections from lower-levels according to their respective scope of impact, and to reach a more accurate detection decision.

To support the objective of a scalable intrusion detection capability, it is important to include the topological information on all the routers affected as part of the scope of impact data. One reasonable representation will be a graph that identifies the routers as nodes, inter-router links as edges, along with the unique identifiers for the routers. In addition, we can also include information such as OSPF adjacency in the data structure. Such information seems useful for making intrusion detection decisions in general. If we allow the possibility of topological information being compiled from a different time than the detection decision, we also need a timestamp associated with the topology graph. Otherwise, the timestamp associated with the detection decision should be sufficient.

4.1.6 Management Information Base (MIB)

JiNao Management information base (JiNaoMIB) is a standard abstraction interface between the JiNao agent and the management applications that are interested in utilizing the intrusion detection services provided by JiNao. The management applications, which will be discussed in the next section, will interact with this MIB through an SNMP engine. This engine will receive and process SNMP PDUs and forward them appropriately to different MIBs. It is also possible for the SNMP engine to communicate the MIB module through so-called *agent extension* protocols. In other words, a MIB access request, as shown in Figure 1, will first pass through a SNMP channel and then pass through the agent extention channel

before it can get to the target MIB. Currently, at least three agent extension protocols have been defined: SMUX[8], DPIv2[9], and more recently AgentX[10].

JiNaoMIB includes five different sections:

Rule/FSM Configuration: Rules and FSMs are used in prevention module as well as local detection module, and they are dynamically loadable from a trusted remote management application. Therefore, the JiNaoMIB specification provide an interface to support this feature.

Each rule in the prevention is represented as a MIB table entry with a set of attributes. By accessing these attributes, we can activate and deactivate the rule. It is also possible to adjust the thresholds used by the rules. All these rule table-entries are placed in a table called "JiNaoPrevRuleTable." Similarly, we will have a table called "JiNaoDetectFSMTable" for all the FSMs used by the local detection module.

Please note that SNMP does not support "direct" table entry insertion and deletion. We will use another MIB variable to indirectly achieve these two unsupported operations. Furthermore, we are expecting to use a secure version of SNMP (like SNMPv2* or SNMPv3/ng). Thus, we need to worry about not only authentication and integrity but also access control. Most of the MIB variables should be restricted to trusted security managers. We should not even allow normal users having read-only access to the rule/FSM tables. If we allow this to happen, then potential intruder will know what attacks JiNao is trying to prevent/protect against, so they can avoid being detected. Therefore, all access to the JiNao MIB must be authenticated and encrypted.

Local Detection Results: The local decision module, after performing analysis on the events or messages, makes decisions about whether certain intrusion attacks have happened. This information is very valuable and should be accessed by trusted security management applications through the MIB interface.

Each piece of information should be represented as a JiNao report table entry (*i.e.*, JiNaoReportTableEntry). All the reports together form the JiNaoReportTable. This table is updated by JiNao decision module periodically to reflect the health of the neighbor routers in real-time. This table is "read-only" through the SNMP/DPIv2 interface. Again, access to the information in JiNaoReportTable must use an authenticated and encrypted message channel with appropriate access control.

Detection Notifications: A particular trusted security management application might be interested in knowing if one particular type of report has been updated in the MIB. Traps/Event notifications are very useful in this situation. In JiNao, this security management application can express its interest in certain types of information through the SNMP MIB interface. The JiNao agent, upon receiving the request, will start to generate traps/events for the application when an event occurs.

Security Control: Most of the security control actions can be achieved by inserting or deleting the rules or FSMs through the rule/FSM configuration MIB section. However, there are cases where we must provide other control interfaces to achieve the goal. For example, a security control console will be directly connected to the local decision

module. A trusted system administrator will use the console to access the JiNao information and control the system directly. Sometimes, a remote management module might want to directly notify the administrator through the MIB interface.

Log Access: In the prevention module, selected PDUs are logged in an audit trail. These logged PDUs may be valuable for some off-line analysis, and they should be accessed through the SNMP protocol.

Apparently, it is unrealistic to provide a unique object identity for each individual record. A search/query engine directly working on the log database is necessary. We merely use the SNMP MIB interface to control this search/query engine. For example, if a management application is interested in receiving all the OSPF PDUs originated by a particular router, then it will, through the SNMP SET/GET interface, submit a query to the search engine. The search engine will retrieve all the matched entries and put them in a table called JiNaoLogAccessResultTable. Then, the requesting security management application can use SNMP Get/GetNext/GetBulk to retrieve all the records in that table.

4.2 Management Information Exchange Protocol

For interoperability, we chose SNMP as the management information protocol to exchange control/management information among the distributed entities in the JiNao system. The current standardized version of SNMP is still version 1 which is not secure (or security was not a concern in version 1). There are at least two proposals for SNMPv2 security: SNMPv2* and SNMPv2u. The new IETF working group: SNMPng (Simple Network Management Protocol: Next Generation) was just established in March 1997. The mission of this working group is to unify different security proposals and to come out with one simple and secure SNMP framework.

The current framework supports two levels of security: message-level security and local process access control. The former concerns a secure (Authentication, Integrity, and Privacy) channel between the MIB agent and an authenticated user. The latter is for capability and resource access control. For example, a normal user's SNMP request for removing a route entry will pass the message-level security, but will be denied by the access control mechanism in local processing module (LPM).

The current security framework of SNMPv3/ng only covers the SNMP PDUs themselves. It does not cover the security concerns for subagent protocols like SMUX, DPIv2, and AgentX. The rationale for this is that the security checks would have been performed by the master agent in the SNMP level. This rationale is fine if the master and subagents are on the same node running a secure OS or both located in a private network segment. However, if it is connected through a public network, the security is an important consideration. For example, we can use any secure transport layer protocol to secure the channel between the master and subagent.

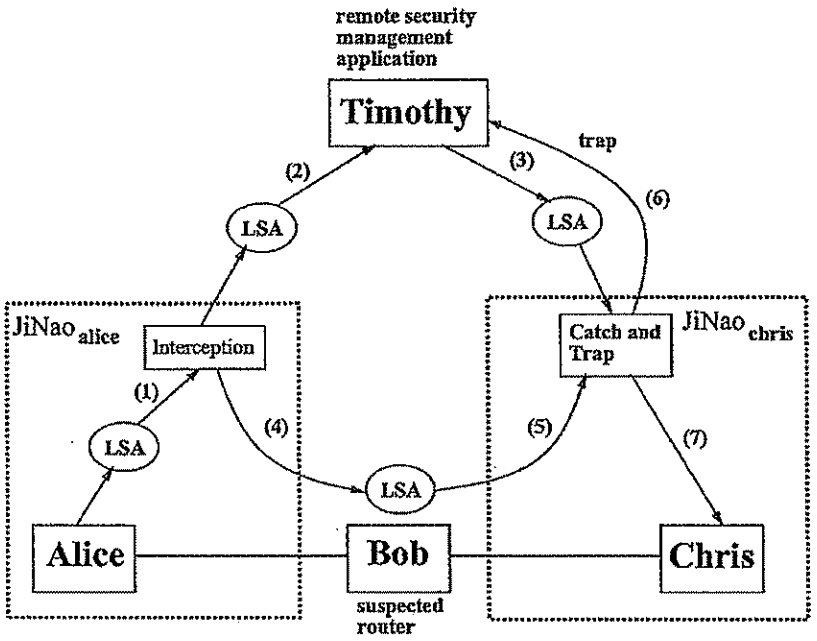


Figure 4: Figurative description of catch and trap.

4.3 Remote JiNao Management Applications

Through the SNMPv3/ng interface, the JiNao IDS service is available to all authenticated and authorized SNMP-based application entities over the public Internet. A remote management station can access the JiNao MIB on different routers and correlate these distributed JiNao results. A MIB specification will be defined such that the remote management applications can interpret the MIB information correctly. In this section, we give an example of remote intrusion detection with a Catch-and-Trap MIB interface.

A remote JiNao management application would like to find out if one particular router is not processing the input protocol data units faithfully. For example, a compromised OSPF router modifies the LSA (Link-State information) originated by this router. If a digital signature scheme is NOT used, it is hard to detect such an attack by one single JiNao. One way to detect this intrusion is to compare the input LSA with output LSA from this compromised router. This can be done by two different approaches using the JiNao MIB interface we just described:

JiNao Log Access MIB: By delegating the proper rules into the prevention module, various types of OSPF PDUs can be logged and retrieved from the Log Access MIB interface. Please note that the prevention module can not only log the incoming PDUs but also the outgoing ones. A remote management application can access the logs on two neighbors of this suspected router. Then, by comparing the log files, the remote management model can tell whether the LSAs have been faithfully forwarded.

JiNao Catch and Trap MIB: Checking and comparing log files might take certain amount

of time, communication and computation resources because the log could contain a large amount of information. Ideally, if the comparison task is performed in the prevention module itself, it will be much more efficient.

The following example is used to describe the functionality of the catch and trap interface: *Alice*, *Bob*, and *Chris* are routers connected to one another as shown in Figure 4. The remote management application *Timothy* is suspecting that *Bob* has been compromised. *Timothy* will send a *suspend* request to the out-going prevention module of *Alice* to catch and hold one outgoing LSA (LSA_x, which should be sent to *Bob*). Now, *Timothy* will use the Catch and Trap interface on *Chris*. After the request, *Chris* knows that he should look at all the OSPF PDUs from *Bob* and check if one of them is LSA_x. At this point, *Timothy* will notify *Alice* to release LSA_x. Now, if *Chris* catches LSA_x, he will trap/notify *Timothy* immediately. If, after δ amount of time, he can not find LSA_x, he will also notify *Timothy* with a Catch-failure report. This catch-and-trap MIB interface facility can be used to efficiently handle compromised routers.

References

[1] J. Anderson, "Computer Security Threat Monitoring and Surveillance", Fort Washington, PA: James P. Anderson Co., April 1980.

[2] J. Winkler, "A UNIX Prototype for Intrusion and Anomaly Detection in Secure Networks", Proceedings, 13th National Computer Security Conference, Oct. 1990.

[3] D. Anderson, T. Frivold, and A. Valdes, "Next-generation Intrusion Detection Expert System (NIDES), A Summary", Technical Report SRI-CSL-95-07, Computer Science Laboratory, May 1995.

[4] D. E. Denning, "An Intrusion-Detection Model", IEEE Transactions on Software Engineering, Vol. 13, No. 2, Feb. 1987.

[5] R. Jagannathan and T. Lunt, "System Design Document: Next Generation Intrusion Detection Expert System (NIDES)", SRI report, SRI International, Menlo Park, CA, March 9, 1993.

[6] L. T. Heberlein, *et al.*, "Internetwork Security Monitor: An Intrusion-Detection System for Large-Scale Networks", Proceedings, 15th National Computer Security Conference, Oct. 1992.

[7] K. Jackson, D. DuBois, and C. Stallings, "An Expert System Application for Network Intrusion Detection", Proceedings, 14th National Computer Security Conference, Oct. 1991.

[8] M. Rose, "RFC1227: SNMP MUX Protocol and MIB", May 23, 1991.

[9] B. Wijnen, G. Carpenter, K. Curran, A. Sehgal, G. Waters, "RFC 1592: Simple Network Management Protocol Distributed Protocol Interface Version 2.0", March 3, 1994.

[10] M. Daniele, B. Wijnen, and D. Francisco, "Agent Extensibility (AgentX) Protocol, Version 1", Internet Draft, Nov 26, 1996.

[11] H. S. Javitz and A. Valdes, "The NIDES Statistical Component Description and Justification", Annual Report A010, SRI, March 7, 1994.

[12] "Wisdom and Sense Guidebook", Los Alamos National Laboratory, Los Alamos, New Mexico.

[13] S. E. Smaha, "Haystack: An Intrusion Detection System", Proceeding IEEE Fourth Aerospace Computer Security Applications conference, Orlando, FL, Dec. 1988.

[14] R. Cleaveland, J. Parrow, and B. Steffen, "The Concurrency Workbench: A Semantics-Based Tool for the Verification of Finite-State Systems", ACM Transactions on Programming Languages and Systems, Vol 15, No. 1, pp 36-72, Jan. 1993.